Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Система аутентифікації за голосом для розумному будинку
Назва теми

КвРКІ. 180117.18.01.14 ПЗ
Шифр

Галузь знань 12 «Інформаційні технології»
Шифр, назва

Спеціальність 123 «Комп'ютерна інженерія»
Шифр, назва

Освітня програма «Комп'ютерна інженерія»
Назва

Виконала: студентка IV курсу, група КІ–18–1 _____ К. В. Россола
Підпис / Ініціали, прізвище

Керівник _____ О.В. Бармак
Підпис, дата / Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата / Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис / Ініціали, прізвище

«1» червня 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 11 " 01 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Россолі Ксенії Вікторівні

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Система аутентифікації за голосом для розумного будинку

Керівник проекту (роботи) Бармак О.В., д.т.н., проф.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2022 р. № 11

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2022 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження поставленої задачі та голосова біометрія

Проектування системи голосової аутентифікації в розумному будинку

Створення програмно-апаратної підсистеми та її тестування

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Інтерфейс програми

Блок-схеми програми

Схема апаратних з'єднань приладу

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання пр
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «_11_»_____01_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Пр
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	11.01.2022	вик
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2022	вик
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2022	вик
4	Робота над розділом 2 – проектування системи голосової аутентифікації	01.04.2022	вик
5	Робота над розділом 3 – створення програмно-апаратної підсистеми	30.04.2022	вик
6	Оформлення пояснювальної записки згідно вимог	20.05.2022	вик
7	Попередній захист ВКР	24.05.2022	вик
8	Захист ВКР на засіданні ЕК	Червень 2022 року	

Студент _____ К.В. Россола
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О. В. Бармак
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 180117.18.01.14 ПЗ	Пояснювальна записка	56		
			Графічні матеріали			
2		КвРКІ 180117.18.01.14 Е8	Інтерфейс програми	1		
3		КвРКІ 180117.18.01.14 Е8	Блок-схема програми	1		
4		КвРКІ 180117.18.01.14 Е2	Схема апаратних з'єднань приладу	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 180117.18.01.14 ВП		
Розробив		Россола			Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Бармак				У	1	1
Н. контр.		Лисенко				ХНУ, КІ-18-1		
Затв.		Говорущенко						

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система аутентифікації за голосом для розумного будинку».

Автор роботи: Россола Ксенія Вікторівна.

Керівник роботи: Бармак Олександр Володимирович.

Пояснювальна записка: 59 с., 51 рис., 8 табл., 4 дод., 31 джерела.

Графічна частина: 8 презентаційних слайдів.

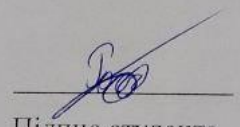
ГОЛОСОВА БІОМЕТРІЯ, ГОЛОСОВА АУТЕНТИФІКАЦІЯ, РОЗУМНИЙ БУДИНОК, ПРИХОВАНА МАРКОВСЬКА МОДЕЛЬ.

Метою роботи є створення системи для аутентифікації за голосом для розумного будинку.

Об'єктом дослідження є програмно-технічний (апаратний) засіб – пристрій для голосової аутентифікації для голосового будинку.

Предметом дослідження є формалізований опис та схеми пристрою для аутентифікації за голосом.

Практичне значення має змодельований, спроєктований та реалізований пристрій на основі голосової біометрії, який використовується для визначення вірного голосу для голосової аутентифікації.



Підпис студента

01.06.2022

Дата

ЗМІСТ

Зм.	Арк.	№докум.	Підпис	Дата	КвРКІ 180117.18.01.14 ПЗ		
Виконав		Россола К.В.			Система аутентифікації за голосом для розумного будинку. Пояснювальна записка	Літера: у	Аркуш:	Аркушів:
Перевір.		Бармак О.В						
Н.контр.		Лисенко С.М.				ХНУ КІ-18-1		
Затвер.		Говорущенко Т.О.						

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АЧХ - Амплітудно-частотна характеристика

MFCC - Mel-frequency cepstral coefficients

GMM – Gaussian Mixture Model

DNN – Deep Neural Network

JFA – Joint factor analysis

ISV – Inter-Session Variability Modelling

I-vector – Total Variability Modelling

SVM – Support Vector Machines

HMM – Hidden Markov Models

PLDA – Probabilistic Linear Discriminant Analysis

ШІМ - Широко-імпульсна модуляція

ПЗ – Програмне забезпечення

ВСТУП

Розумний дім — система домашніх пристроїв, здатних виконувати дії і вирішувати певні повсякденні завдання без участі людини.

Розумний дім дуже зручний, тому що це звичайний будинок або квартира, але має в собі «розумну» систему, яка виконує будь-яку забаганку та будь-яке бажання свого господаря і сама вирішує безліч побутових завдань. Жити в такому будинку не лише втішно, а й зручно, безпечно та навіть вигідно.

В розумному будинку є багато усіляких функцій. Є надійна і проста система охорони та відеонагляду, яка легка у користуванні; автоматичне підлаштовування освітлення, яке залежить від часу доби, яке полегшить пересування в приміщенні (особливо важливе воно, коли в будинку є діти або люди похилого віку); контролювання за протіканням газу та води. Також може допомагати з рутинними справами, наприклад відкрити двері для собаки о 8 ранку; віддалено керувати будинком та побутовими приладами за допомогою телефону чи інтернетом, щоб, наприклад через 5 хвилин підігрів чайник до вашого повернення додому. Увесь будинок допомагає покращити умови для проживання та виконання побутових задач, а особливо для інвалідів та для старих людей. І все це можна виконувати за допомогою єдиного пульта-дисплея. Але найголовніше — безпека будинка. І її можна досягти тим, що можна розмістити датчик, який буде зчитувати голос для аутентифікації, щоб мати доступ до будинку.

ДОСЛІДЖЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ ТА ГОЛОСОВОЇ БІОМЕТРІЇ ДЛЯ РОЗУМНОГО БУДИНКУ

1.1 Загальна інформація

У сучасному світі все частіше і частіше проявляється цікавість до технологій ідентифікації за допомогою людського голосу, особливо для розумного будинку. З одного боку це пояснюється реалізацією високопродуктивних систем які можуть рахувати складні сигнали, наприклад, голос.

Як правило, звуковий зразок обробляється блоками, тобто спочатку записується частина фрази по заданій довжині, а потім обробляється та аналізується вся фраза. Чим більша кількість аналізованих фрагментів звуку, тим довший час перевірки голосу. Використання розпізнавання в режимі реального часу вимагає швидких і економічно ефективних алгоритмів обробки сигналів, які можна встановити навіть на невеликих комп'ютерних пристроях.

Розпізнавання голосу має дві основні підгрупи: розпізнавання мовлення (розпізнавання тексту по голосу) та голосову біометрію (розпізнавання по голосу особу). Для виконання цієї дипломної роботи використовується друга підгрупа - голосова біометрія.

Також є 4 типи системи розпізнавання голосу [1], які показані нижче на рисунку 1.1.

1) ізольована система розпізнавання голосу - система вимагає короткого проходу між вимовленими словами;
2) система безперервного розпізнавання голосу - як випливає з назви, ця система не вимагає жодного проходу між словами;
3) система розпізнавання голосу, залежна від голосу - ця система розпізнає мовлення лише одного динаміка. Це означає, що тільки певний оратор може пройти через цю систему;
4) незалежна система розпізнавання голосу - ця система може розпізнати мовлення будь-якої людини.

Рисунок 1.1 – Класифікація систем

Біометрія голосу є однією з технологій, що швидко розвиваються, що дозволяє різним компаніям використовувати свої рішення для ідентифікації клієнтів. Біометрична система ідентифікації або аутентифікації з використовує особисті, психічні та інші характеристики. Існує багато біометричних вимірювань, зокрема сканування райдужної оболонки ока, відбитків пальців, розпізнавання обличчя, звуків, підписів тощо. Biometrics Voice дозволяє ідентифікувати клієнта, дивлячись на характеристики голосу людини. Це простий спосіб фінансово вирішити ряд значущих проблем.

Біометрія голосу та мовна технологія більше не є іграшками, це високорозвинена технологія, яку можна використовувати для покращення якості обслуговування до такої міри, щоб клієнти відчували це покращення. Компанія повинна надати клієнту пряму послугу, і мовні технології можуть допомогти. Ніхто не змушує клієнта чекати, перенаправляти або відображати використання меню. Голосовий зв'язок зручний для клієнтів.

Система розуміє клієнта і вміє переглянути на його слова. Він може навіть не пам'ятати ключа чи номера. Біометричні пристрої, що використовуються для зв’язку, дозволяють ідентифікувати користувача. Це зменшує час розмови. Тому клієнту не потрібно представлятися і називати пароль. Його пароль - це його голос! Водночас він відчуває, що його дзвінок важливий, і компанія одразу приймає рішення.

Біометричний метод розпізнавання голосу може свідчити про обов'язковість подальшої перевірки власника. Ви також можете створити «чорний список» відбитків голосів людей, виявлених у результаті шахрайства або спроби отримати несанкціонований доступ до будинку.

Голосова біометрія — це тип перевірки клієнта, за допомогою якого можна визначити, чи жива людина чи оприлюднюється запис.

Треба зазначити, що розпізнавання голосу розбігається від розпізнавання мовлення. Хоча технологія розпізнавання голосу розтлумачує те, про що говорить абонент, технологія розпізнавання голосу застосовується для ідентифікації мовця. Оскільки голос можна легко записати на плівку або інший пристрій, деякі виробники включають відповідь на запит програми у свій продукт.

Ця функція спонукає користувача відповісти на заздалегідь запрограмований запит, який змінюється під час введення, наприклад: «повторити цифри 0, 3, 5». Біометричний метод розпізнавання голосу простий у використанні. Однак найбільшим безумовним недоліком цього методу є низька точність ідентифікації.

Дійсно, технологія розпізнавання голосу має свої обмеження. Різні люди можуть говорити однаковими голосами, і голос кожного може змінюватися з часом залежно від стану здоров'я, емоційного стану та віку. Наприклад, людині з грипом або ларингітом може бути важко використовувати ці методи. Якість з'єднань можуть ускладнити ідентифікацію [2].

У розумному домі з голосовою біометричною системою процес поділяється на дві основні частини, це процес розробки програмного забезпечення та апаратного забезпечення.

Розробка програмного забезпечення включає збір даних, обробка аудіосигналу та розробка графічного інтерфейсу користувача (GUI). Тим часом, розробка апаратного забезпечення включає проектування прототипу та створення схеми.

1.2 Проблеми голосової аутентифікації, її плюси та мінуси

Зараз голосова біометрія стала майже такою ж точною, як і інші види біометричних аутентифікацій. Можна припустити, що звук можна легко записати на диктофон і відтворити, коли потрібно пройти аудіотест, увійшовши таким чином у несправність системи. Так, дійсно, не всі системи захищають це. Більше того, за статистикою Pindrop, рівень голосового шахрайства зріс на 350% у 2013-2017 роках без ознак зниження. При цьому голосове шахрайство також зросло на 47% у 2016-2017 роках, тобто кожен із 638 дзвінків виявився фейковим.

Графік величини голосового обману за 2013-2017 рр. можна побачити на рисунку 1.2.

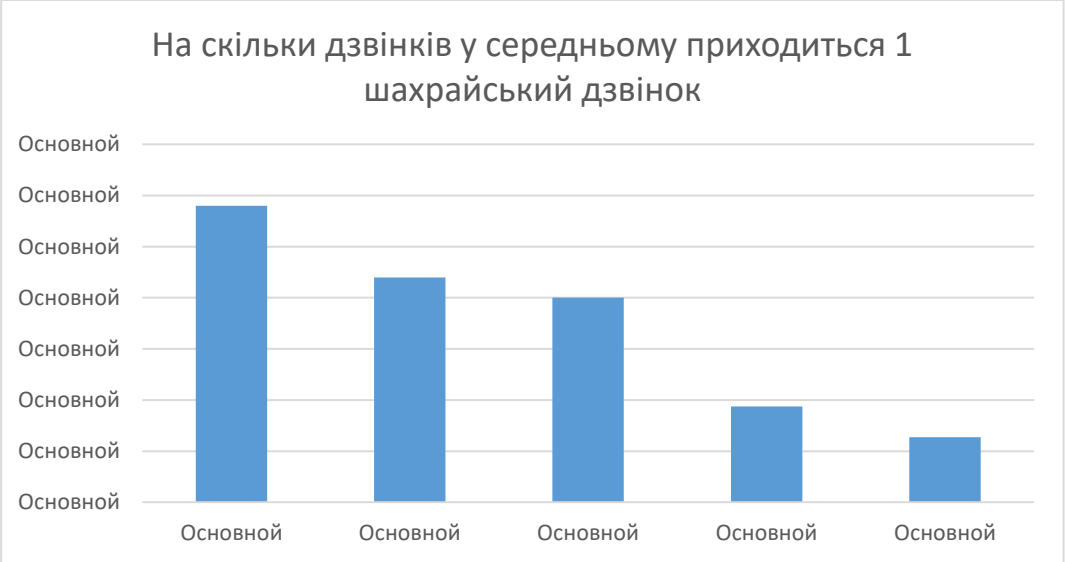


Рисунок 1.2 — Графік величини голосового шахрайства у 2013-2017 рр.

Pindrop розпитав 500 керівникам бізнесу та IT. Були опитані компанії зі США, Франції, Великобританії та Німеччини. Протягом травня 2018 року Loadhouse провела онлайн-опитування.

В порівнянні з паролями, кодами, ключами та картками, голосова аутентифікація має такі переваги:

- так як це наш голос, то його неможливо втратити чи забути;
- ми не можемо передати комусь наш голос;

- істотно важко підробити наш «голосовий ключ»;
- велика зручність.

Але також у нас є і мінуси:

- зміна голосу (хвороба, нещасний випадок);
- запис нашого голосу чужим пристроєм.

У таблиці 1.1 показано ймовірності невдач, так як усе не може бути ідеальним.

Таблиця 1.1 – Ймовірності невдач

Ймовірність відмови у доступі, %	Ймовірність помилкової ідентифікації «чужого» (без використання муляжу)	Ймовірність помилкової ідентифікації «чужого» (з використання муляжу)	Збереження таємниці образу у процесі ідентифікації абонента	Вартість технічної реалізації в грошовому еквіваленті, у.о.
0,5…5	0,5…5	25…90 (запис)	10-16…10-30	1…60

1.3 Принцип роботи голосової аутентифікації

В системі встановлений мікрофон. Їх є безліч видів, але усі вони працюють за одним принципом. Звукова хвиля потрапляє на мембрану, де вібрації мембрани передають на еластичний елемент, який перетворює електричні коливання в електричні сигнали. Сигнал посилюється і вставляється на вхід звукової карти. Звукова карта є аналого-цифровою альтернативою перетворювача. Найважливішими параметрами є частота дискретизації та розрядність кодування. Ці фактори безпосередньо впливають на якість запису, а в результаті і на розмір самого запису [3]. Система розпізнавання голосу працює за такою процедурою:

- створюється реєстрація користувача та виконуються розрахунки структури;

- для аналізу обираються часові діапазони мовного потоку;
- реалізується базова обробка сигналів;
- зчитуються первинні параметри;
- створюється шаблон голосу;
- голосовий шаблон порівнюється з тим, який вже є в базі.

Процес зчитування голосу та зберігання в базі називається Enrollment.

Процес порівняння голосових шаблонів складається з таких кроків:

- фільтрація шумів;
- спектральне перетворення сигналу;
- пост фільтрація спектра;
- ліфтеринг;
- накладання вікна Кайзера;
- порівняння.

Сам процес зчитування голосу та порівняння його з правильним називається Test.

При підборі шматочків фрагментів використовуються різні методи. Ви можете використовувати весь потік мовлення, крім пауз. Ви також можете вибрати фрагменти найбільш шумних ділянок, оскільки ймовірність шуму найменша. Також можна вибрати голосні звуки, оскільки вони можуть визначати характер вимовляння фрази, тощо.

На рисунку 1.3 показана ймовірність існування специфічних ознак голосу людини в 16 фонемах.

Фонема	є	о	л	а	і	з	р	в	ж	м	г	ч	ц	с	ш	к
Ймовірність	0,9	0,86	0,84	0,83	0,83	0,79	0,78	0,76	0,62	0,62	0,61	0,54	0,5	0,44	0,37	0,3

Рисунок 1.3 – Ймовірність існування ознак в голосі в 16 фонемах

У процесі обробки перших сигналів аналізуються спектральні параметри мови. Основним методом є фільтрація вузьких сигналів. При оголошенні

остаточного вислова сигнал приходить до одного масштабу амплітуд за рахунок підсилювача. Перші ознаки сигнала мають такі характеристики:

- не залежить від шумів;
- мало піддаватися імітації;
- легко виділяються із сигналу;
- відображення індивідуальності диктора;
- бути незалежними до фізичного та емоційного стану диктора.

Базові параметри можуть використовувати АЧХ, фон, інтервал між високими звуками, афікс, тривалість окремих звуків, тощо. Під час вимовляння пауза між звуками може змінюватися в межах 10 - 50%. Щоб компенсувати цю нестабільність, можна використовувати такі методи:

- стиснення або розтягування окремих ділянок;
- виділення центру звукової зони, тоді виміри навколо центру не відіграють сильної ролі.

На рисунку 1.4 показано, як виглядає схема введення голосу.

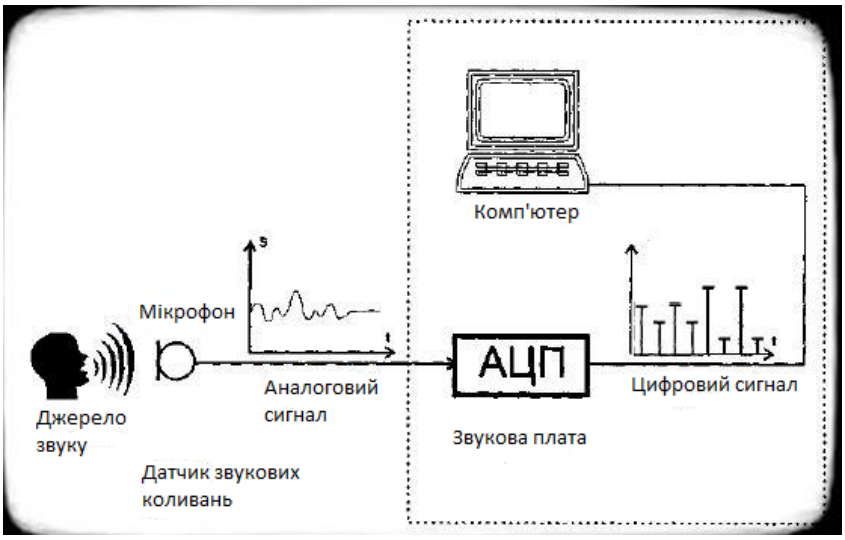


Рисунок 1.4 – Схема введення голосу

А на рисунку 1.5 можемо побачити загальну схему зчитування голосу, зберігання в базі та порівняння.

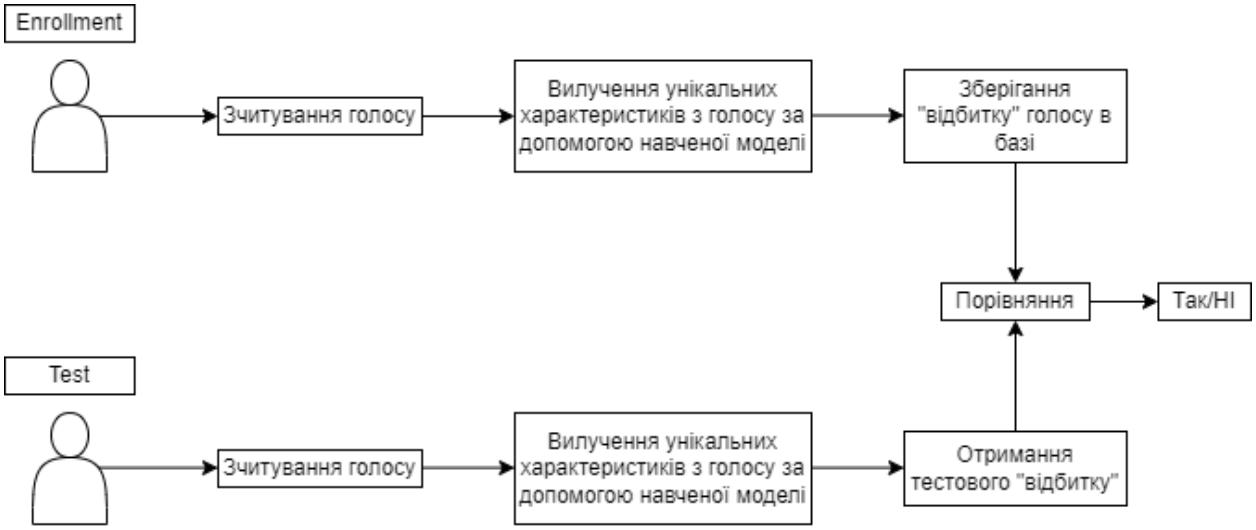


Рисунок 1.5 – Загальна схема аутентифікації

Момент, коли система зчитує голос, називається передобробка.

Є багато різних способів як покращити сигнал перед тим, як вилучати з нього ознаки в зокрема накладання різних фільтрів, таких як Pre-emphasis або band pass фільтр.

Сигнал являє собою великий масив чисел в якому в кожній секунді відповідають числа елементів рівною частоті дискретизації нашого сигналу.

Щоб застосувати Pre-emphasis нам потрібно з кожного елемента цього масива відняти попередній помножений на якийсь коефіцієнт, як правило 0,95 або 0,97.

Це допомагає нам підвищити амплітуди ділянок з високою частотою і понизити для ділянок з низькою частотою.

Шум має відносно низьку частоту порівняно з голосом людини, тому така процедура допомагає покращити відношення сигналу і шуму в записі [4].

На рисунку 1.6 показано приклад застосування фільтру Pre-emphasis.

Наступний важливий етап це фільтрація шумів.

Звук, який створений за допомогою коливаннями всього діапазону частот називається шумом і його спектр показаний на рисунку 1.7.

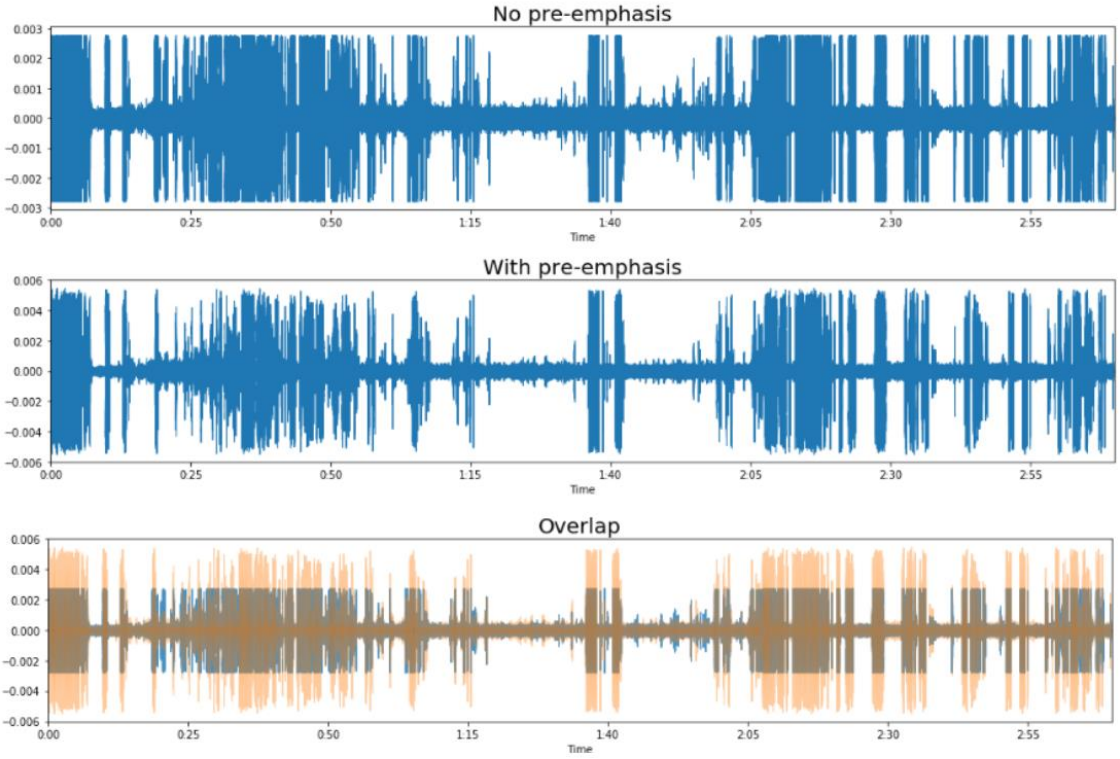


Рисунок 1.6 – Приклад застосування Pre-emphasis

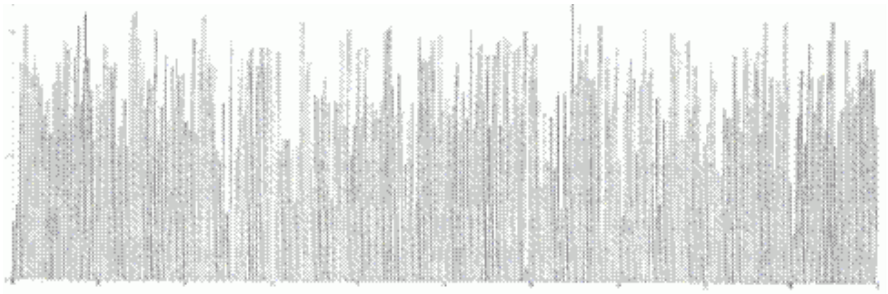


Рисунок 1.7 – Спектр шуму

Для того, щоб мати змогу отримати спектральні характеристики звука, то потрібно очистити від лишніх шумів. Він оброблюється фільтрами, щоб позбутися від перешкод.

$$X_i = (X_i - 0{,}9 * X_{i-1})[0{,}54 - 0{,}46 * cos\ cos\ ((i-6) * \frac{2*\pi}{180})\ ], \quad (1.1)$$

де X_i – набір дискретних значень звукового сигналу.

Щоб знайти початок і кінець фрагментів в масиві дискретних значень сигналу, то так як шуми вже відфільтровані, то X_0 буде початком фрагменту і буде характеризуватись сплеском сигналу, а X_n буде характеризуватись кінцем фрагменту. Якщо говорити не в математичному вигляді, то знайшли слово, яке сказала людина і потрібно порівняти з іншими характеристиками голосу.

Є таке як пікова амплітуда. Це абсолютна величина максимально отриманих з дискретних значень рівня звука.

І для того, щоб уникнути спотворення, яке залежить від меж сигналу, то треба звернути увагу на цю амплітуду. І при цьому треба зберігати відношення сигналу/шуму при максимальному рівні.

Електричні сигнали, що йдуть по кабелям, передають потужність. По кабелям звук передається у вигляді змінної напруги і ця потужність дорівнює звуку пропорційна квадрату напруги. Щоб порахувати повну потужність за період часу, треба просумувати всі значення моментальної потужності цей період. Це можна зробити за допомогою формули:

$$\int v_t^2 dt, \tag{1.2}$$

де v_t – напруга в заданий час.

За допомогою методу СКЗ визначається середнє для швидко мінливої величини. Завдяки цьому, можемо пов’язати середню амплітуду та та середню потужність, так як миттєва потужність залежить від квадрату миттєвої амплітуди. СКЗ амплітуди буде:

$$\sqrt{\frac{1}{N}\sum_{i=0}^{N}(x(i))^2}, \tag{1.3}$$

де $x(i)$ – амплітуда і-того дискретного значення.

Так як будь-який звук розкладається на синусоїдальні хвилі, то можна зробити частотний спектр звуку, який представляє з себе графік залежність амплітуди від частоти [5].

Склавши 2 синусоїдальні хвилі з однаковими частотами, то це буде одна, навіть якщо мають різні амплітуди і фази. А для виміру амплітуд одної частоти, потрібно помножити сигнал, який вже є на синусоїд такої ж частоти та скласти отримані відліки. Вимірюється амплітуду частоти f у першому наближенні, при обчисленні наступної суми:

$$A_f = \sum_{t=0}^{N-1}(e^{-2\pi tf/N}), \tag{1.4}$$

де f – ціла кількість, а реальна досліджувана частота – це частота дискретизації, помножена на f/N, t — цілий номер відліку.

Коли взнали A_f, то можна взнати відліки. Але щоб зробити зворотне перетворення Фур'є, ще треба фазу кожної частоти. А для цього треба комплексі числа. Використовуючи комплексні числа, можна проводити вимірювання одночасно, помножуючи синусну частину -i.

$$A_f = \sum_{t=0}^{N-1}(s_t * cos\ cos\left(\frac{2\pi tf}{N}\right) - i * s_t * sin(\frac{2\pi tf}{N})). \tag{1.5}$$

Основна ідея перетворення Фур'є полягає в тому, що кожну другу вибірку можна використовувати для отримання половинного спектру. Формально це означає, що формула дискретного перетворення Фур'є може бути представлена у вигляді двох сум. Перша містить усі парні компонент, друга – усі непарні.

$$A_f = \sum_{t=0}^{N/2-1} s_{2t} e^{-2\pi tf(\frac{N}{2})} + e^{-2\pi tf/N} * \sum_{t=0}^{N/2-1} s_{2t-1} e^{-2\pi tf(\frac{N}{2})}. \tag{1.6}$$

Отримавши спектральне представлення сигналу, його потрібно почистити від шумів.

Людський голос має особливі характеристики, і тому ті області, які не можуть бути характеристиками голосу, потрібно забрати. Для цього використовуємо функцію "вікно Кайзера".

$$y = \frac{x}{2}. \quad I_0(b) = \sum_{n=1}^{50}\left(\frac{I_0(b)_{n-1} * y}{n}\right)^2, i = 1 \dots n, b = 5, k = -\frac{n}{2}. \qquad (1.7, 1.8)$$

$$I_{01} = \frac{1}{I_0(b)}. \quad X_i = X_i\left[b * \sqrt{1} - \left(\frac{2*k}{n-1}\right)^2\right] * I_{01}. \qquad (1.9, 1.10)$$

Після фільтрації спектру накладемо вікно Ханнінга.

$0 \le n \le Ns$; $\alpha_w = 0{,}54$;

$$W(n) = \frac{\alpha_w - (1-\alpha_w) * cos\left(\frac{2\pi n}{N_s - 1}\right)}{\beta_w}. \qquad (1.11)$$

Також потенційно корисний етап передобробки це розділення людей, які говорять (Speaker Diarization). Буває і таке, що при процесі мовлення може хтось стояти поруч та теж говорити, чи хтось просто проходив та щось говорив. В результаті одного запису отримуємо декілька голосів.

Моделі на таких записах не дуже зручно навчати. Можна спочатку розділяти ці голоси, але це не швидко і нетривіально, тому є швидший спосіб, завдяки якому є модель, яка класифікує голоси людей по жіночій/чоловічій статті в залежності від голосу.

На рисунку 1.8 показано приклад схеми розділення голосів.

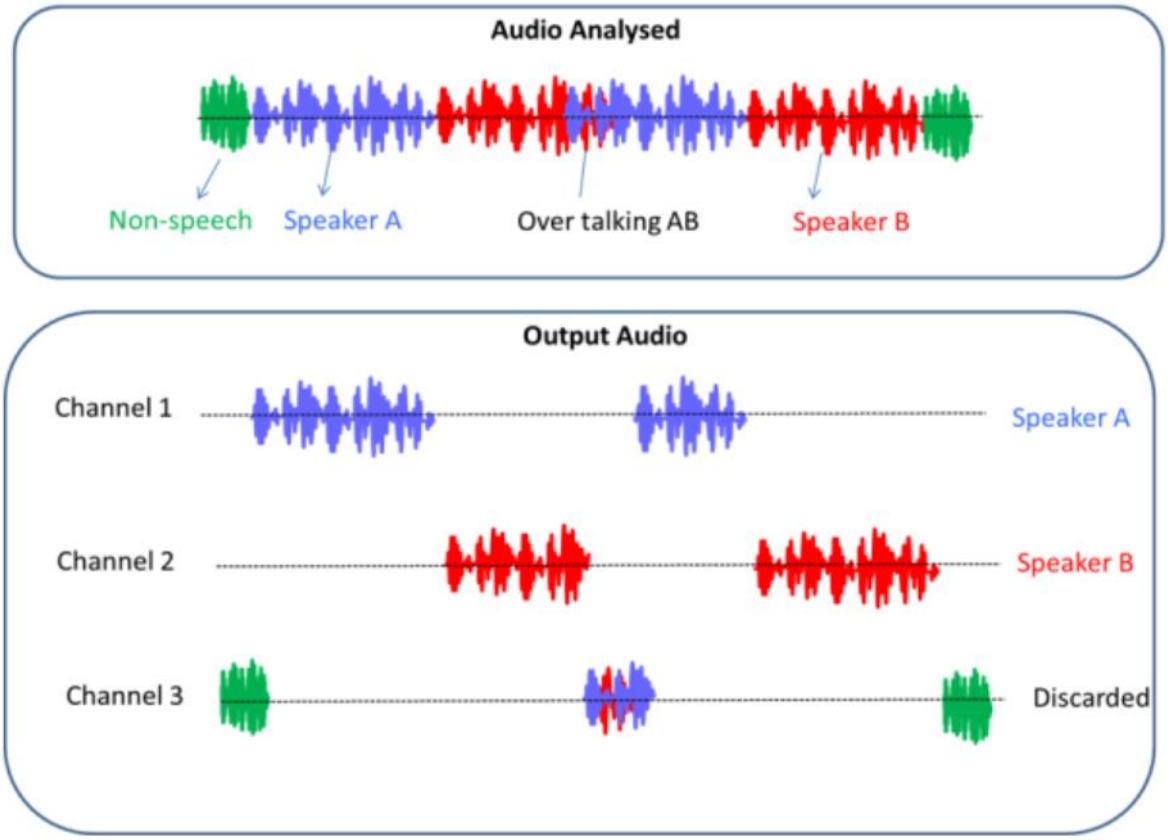


Рисунок 1.8 – Приклад розділення голосів

Далі потрібно вилучити особливі моменти з голосу, які допоможуть визначати вірний голос. Як правило використовуються коефіцієнти MFCC, це достатньо стандартний підхід до вилучення ознак. Йдуть такі кроки [6]:

1) сигнал розбиваються на шматочки по 20-30 мс з перекриттям 10-15 мс, плюс застосовуємо віконну функцію (Hamming window):

$$w[n] = 0{,}54 - 0{,}46\frac{2\pi n}{N-1}; \quad (1.12)$$

2) перетворення Фур'є (перехід від сигналу до енергетичного спектру);

3) перехід до mel-шкали:

$$m = 2595(1 + \frac{f}{700}); \quad (1.13)$$

4) дискретне косинусне перетворення («спектр спектру») для отримання некорельованих ознак;

5) похідні MFCC 1-го та 2-го порядку.

Головна ідея така, що на виході отримуємо ряд коефіцієнтів, які некорельовані між собою і відображають особливості сприйняття людським вухом голос.

Часто до MFCC додають ще динамічні коефіцієнти, тобто точні оцінки похідних MFCC 1-го та 2-го порядку. Ця процедура допомагає понизити помилку верифікації на 20%.

На виході отримуємо прямокутну матрицю, де по одній осі, власне, коефіцієнти, а по-іншій часова шкала. Часто в такому вигляді її подають на вхід нейронним мережам. На рисунку 1.9 зображено приклад того, як виглядає ця матриця.

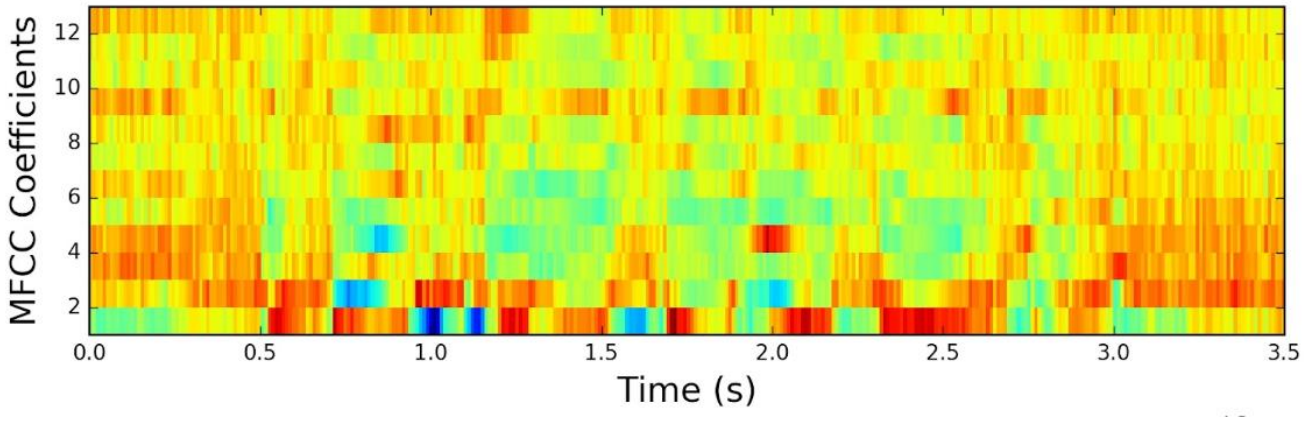


Рисунок 1.9 – Приклад результату матриці після процедури MFCC

Далі потрібно використати моделі, які допоможуть створити «відбитки» голосів.

Зараз найбільш популярні 2 підходи: моделі засновані на моделі гаусової суміші (GMM) та моделі глибокого навчання (DNN). Вони містять в собі такі моделі [7]:

1) GMM-based:
    a) UBM-GMM;
    b) GMM-SVM;
    c) JFA;

d) ISV;

e) I-vector;

2) DNN-based:

a) D-vector;

b) X-vector;

3) Інші:

a) SVM;

b) HMM.

У використанні зручні такі моделі, як I-vector і D-vector.

I-vector представляє собою сумішшю gml моделі основані на GMM (рисунок 1.10) і факторного аналізу (рисунок 1.11).

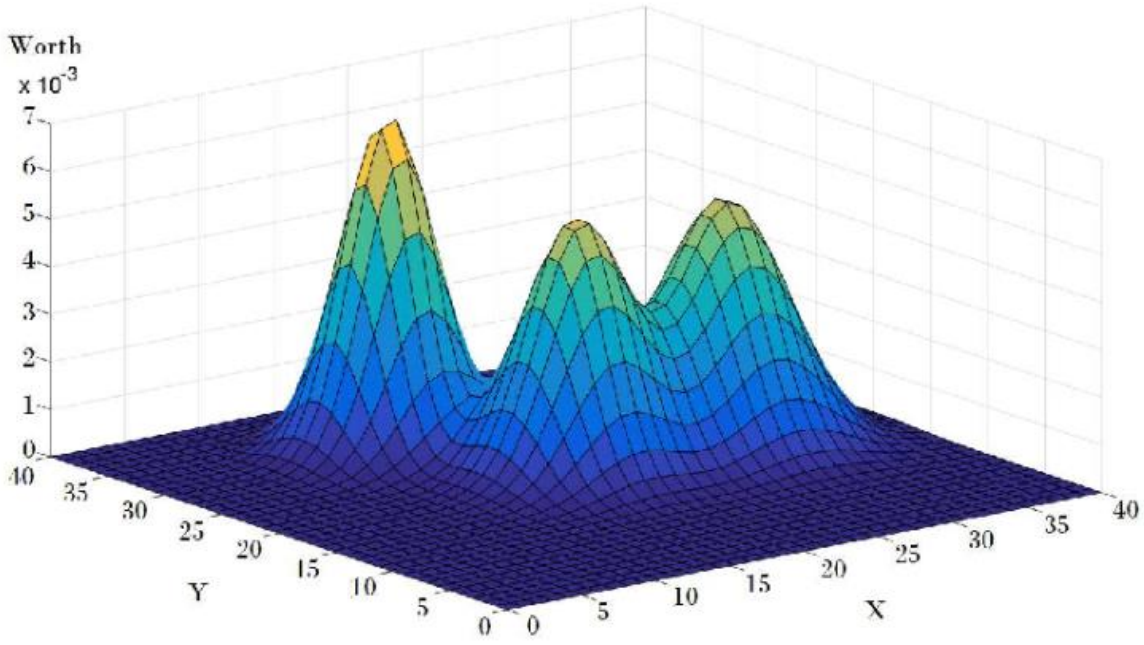


Рисунок 1.10 – Gml модель

Для початку щільність ймовірності на всьому просторі голосових мереж наближається за допомогою якоїсь заданої кількості нормально багатовимірних нормальних розподілів. Як правило, як кілька сотень або навіть тисяч.

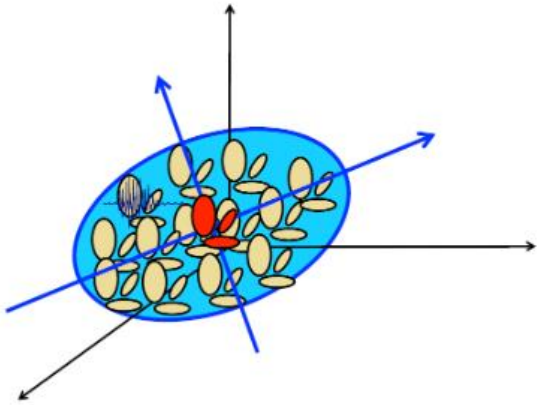


Рисунок 1.11 – Факторний аналіз

На картинці вони двовимірні, хоча, по факту, розмірність збігається з кількістю фічів MFCC, які були до цього вилучені. Кожна компонента суміші представлена в багатовимірній функції щільності ймовірності з відповідним вектором середніх і матрицею коваріацій. Ця суміш називається UBM [8].

$$P(UBM) = \sum_{i=1}^{N} \omega_i P(X|\mu_i, \sum_i), \tag{1.14}$$

де N – число компонентів суміші;

ω_i - вага i-ого компонента;

$\sum_i$ – матриця коваріацій i-ого компонента;

μ_i – вектор середніх i-их компонентів;

D – розмірність вектора X (число ознак MFCC).

$$P(\mu_i, \sum_i) = \frac{1}{\sqrt{(2\pi)^D |\sum_i|}} exp\ exp\ (-\frac{1}{2}(X-\mu_i)^T \sum_i^{-1}(X-\mu_i)). \tag{1.15}$$

Після того, як було оцінено розподілення на усьому просторі, наступна задача є адаптувати вектор середніх всіх гаусових VBM під кожний конкретний запис. Це робиться за допомогою процедури максимуму апостеріорної оцінки (MAP). На виході можемо кожен запис кожного зчитуваного голосу представити у вигляді великого вектора, як правило, який містить кілька десятків тисяч

елементів, який ще називається супер вектор і кількість елементів в ньому дорівнює D*N, де D – кількість наших MFCC, а N – кількість гаусіан в нашій суміші в GMM. Цей вектор вже непогано характеризує голос людини, яка говорить і може бути вже використаним для аутентифікації, він буде нормально працювати, але він все рівно достатньо великий і містить в собі багато лишньої інформації, тому наступна задача – понизити розмірність цього вектора за допомогою факторного аналізу [9-10].

$$M = m + T\omega, \tag{1.16}$$

Де супер вектор M (вектор з GMM) представляється в якості суми компоненти незалежно від людини (m, в якості неї обирають вектор середній із всіх гаусових в нашій універсальній моделі), також $T\omega$, де T – прямокутна матриця невеликого рангу, а ω - латентна змінна, вектор розмірністю від 100 до 1000 елементів, як правило.

Після того, як зробили розмірності та отримали I-vector, часто, непогано було б застосувати методи передобробки, такі як нормалізація довжини вектора, нормалізація всередині класової коваріації лінійний дискримінантний аналіз. Ці всі методи допомагають понизити вплив особливостей каналу, тобто в умовах, в яких був зроблений конкретний запис, таких як шум [11-12].

Далі наступає етап скорінг, при якому маємо отримати значення, яке відображає схожість 2 відбитків еталонного та тестового і видати результат, що це правильна людина чи ні.

Найпростіше, що може сюди підійти, це косинусна міра близькості (cosine similarity), але не дає дуже хороші результати, принаймні на I-vectori. Тому є трішки складніший, але набагато ефективніший метод ймовірнісний лінійний дискримінантний аналіз (PLDA), який дозволяє представити наш вектор у вигляді компоненти залежні від людини ($\mu + Fh_i$) і компонентів, які залежать від навколишнього середовища ($G\omega_{i,j} + \epsilon_{i,j}$).

$$X_{i,j} = \mu + Fh_i + G\omega_{i,j} + \epsilon_{i,j.} \quad (1.17)$$

В підсумку, за допомогою PLDA можемо оцінити відношення ймовірності 2 гіпотез, що гіпотеза H_0 належить до правильного голосу, але були зроблені в різних умовах, а гіпотеза H_1, що різні голоси, але в схожих умовах.

Також є альтернативний підхід до створення «відбитків» людини і він називається D-vector. Для нього можуть використовуватися найрізноманітніші архітектури, але основна ідея полягає в наступному: оскільки згорткові нейронні мережі навчені класифікувати картинки на останніх шарах отримувати важкі представлення, такі як очі та вуха, то вона так само може на голосових мережах на останніх шарах вивчити представлення людини, яка буде дуже добре відрізняти від всіх інших. Власне, всередині цієї нейромережі, як правило, обирається один з останніх прихованих шарів і в залежності від того, який з них краще поділяє людей між собою і вихід цього шару використовується в якості дивектора «відбитку» голосу. На вхід таких нейромереж можуть подаватися як MFCC, так і сигнал в чистому вигляді після передоброки.

Найцікавіший варіант — архітектура SincNet. ЇЇ особливості такі, що на вхід подається сигнал без вилучення фічів, а модель сама вчиться вилучати ознаки. На фільтрах перших шарах накладені певні обмеження їх параметрів з зв’язку з чим модель швидше навчається і менше перенавчається [13-14].

1.4 Прихована Марковська Модель

Також є дуже хороший математичний метод голосового розпізнавання, як прихована марковська модель (Hidden Markov Model, HMM). Ця модель є основою набору успішних методик акустичного моделювання в системах розпізнавання мовлення.

Перше практичне застосування методу HMM для розпізнавання звуку відбулося ще у 1970-х роках, таким чином це дуже опробований підхід, який добре зарекомендував себе саме для розпізнавання звуку.

Основні причини такого успіху пов'язані з аналітичними здібностями цієї моделі в мовленнєвому феномені та її точністю в практичних системах розпізнавання мовлення. Іншою важливою специфікацією HMM є його конвергентна та надійна процедура навчання параметрів. Умовні висловлювання представлені у вигляді нестаціонарної послідовності векторів ознак. Отже, щоб статистично оцінити послідовність мовлення, потрібно сегментувати послідовність мовлення на стаціонарні стани. Модель HMM — це кінцевий автомат. Кожен стан можна моделювати як одну гауссову або мультимодальну суміш Гаусса [14-15].

Плюсом використання HMM є її наочність та можливість роботи із об’ємними інформаційними сигналами. До мінусів можна віднести необхідність уточнення особливостей розподілу випадкових величин, що характеризують виходи моделі, а також те, що алгоритми її навчання зазвичай дозволяють досягти лише локального оптимуму.

HMM досить прості в розумінні і мають досить високу точність розпізнавання. HMM можуть застосовуватися у багатьох сферах, де метою є виявлення послідовності даних, що не є безпосередньо спостережуваною (але інші дані, що залежать від цієї послідовності, спостерігаються). Це насамперед стосується Другої та Третьої проблеми (Decoding та Learning), яку вирішує HMM. Що до застосувань де потрібно порівняти декілька послідовностей, то застосовують Першу та Третю проблеми (Evaluation та Learning). Саме ці дві функції HMM використовувались у курсовій роботі, коли порівнювались голосові шаблони, якими була навчена HMM, з тестовою вибіркою векторів ознак голосових зразків різних персон.

В загалі, до напрявків застосування HMM належать:

- розпізнавання мовлення;

- кінетичний аналіз однієї молекули;
- криптоаналіз;
- синтез мовлення;
- морфологічна розмітка;
- розділення документів у рішеннях для сканування;
- машинний переклад;
- передбачення генів;
- вирівнювання біопослідовностей;
- аналіз часових рядів;
- згортання білків;
- виявлення метаморфних вірусів;
- виявлення консервативних мотивів ДНК.

В основному, HMM є таким набором символів:

$$\lambda = (N, M, A, B, \pi), \tag{1.18}$$

де N – кількість станів моделей;

M – кількість відмінних символів спостереження одного стану;

A = {aij} – NxN матриця розподілу ймовірностей зі стану i в стан j;

B = {bjk} – NxM матриця розподілу ймовірностей спостередження k-го символу у стані j;

π - вектор розподілу ймовірності початкового стану i.

Приклад HMM показаний нижче на рисунку 1.12 (N=2, M=3) [16].

На рисунку 1.13 показано загальну архітектуру HMM.

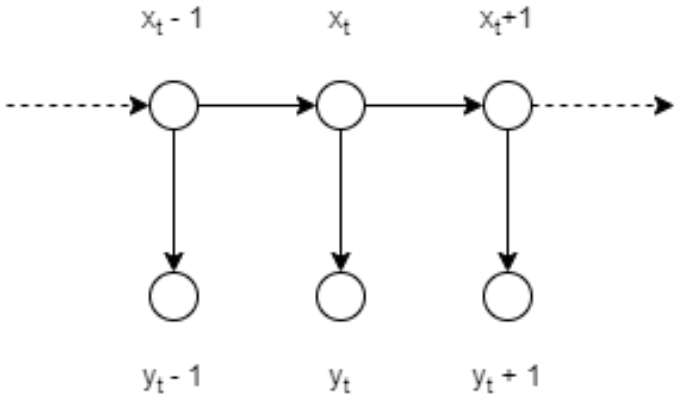


Рисунок 1.13 – Загальна архітектура HMM

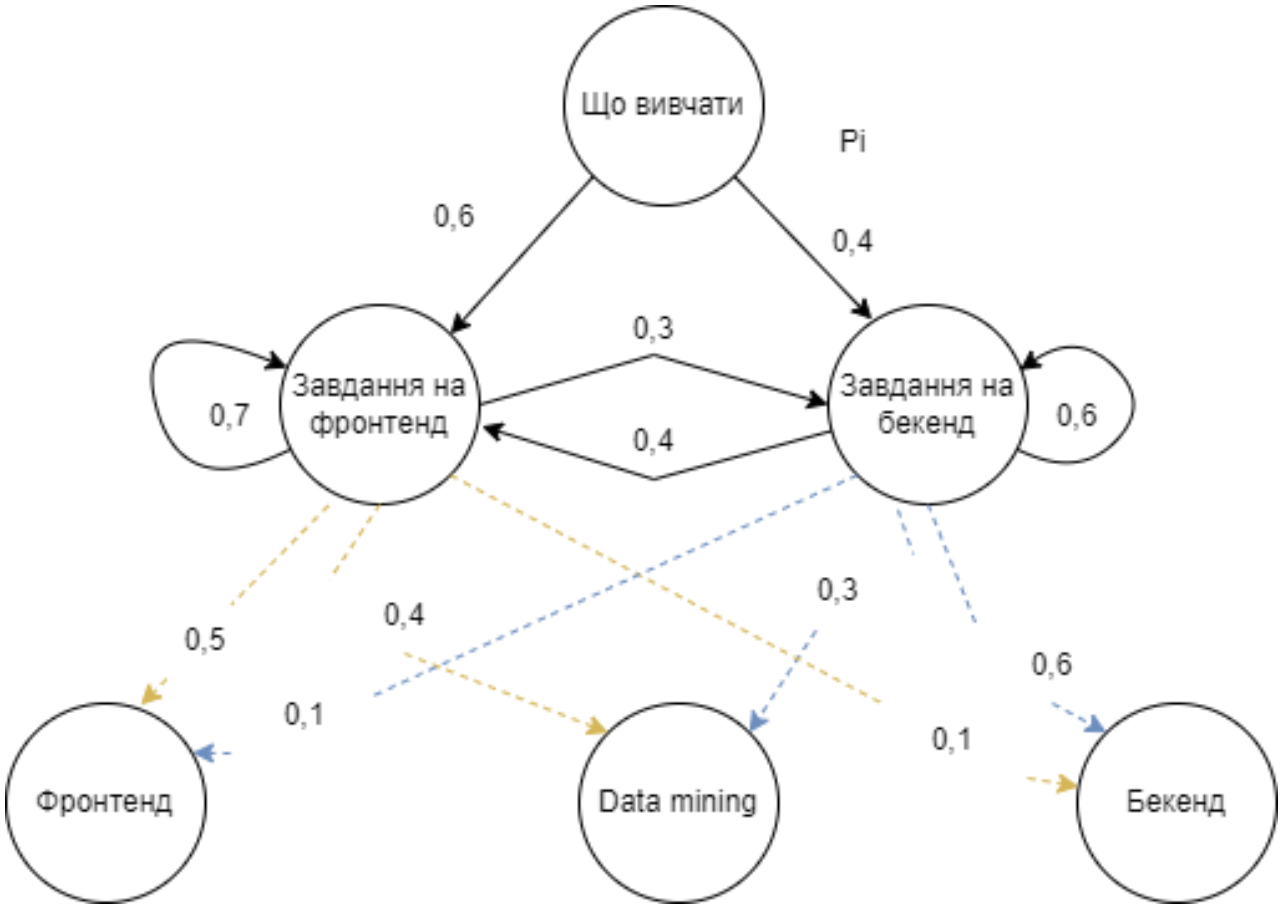


Рисунок 1.12 – Приклад схеми HMM

Кожне коло означає випадкову змінну. Змінна x(t) є прихованим станом в час t. А y(t) є спостередженням у момент часу t. Стрілки показують ймовірність залежності.

1.5 Висновки

У цьому розділі було обґрунтовано про тему голосової біометрії, яка застосовується в даній дипломній роботі, процес проходження голосової аутентифікації, її актуальність і також її плюси та мінуси. На основі аналізу, можна зрозуміти, що буде важке розроблення ПЗ.

ПРОЕКТУВАННЯ СИСТЕМИ ГОЛОСОВОЇ АУТЕНТИФІКАЦІЇ В РОЗУМНОМУ БУДИНКУ

2.1 Обґрунтування вибору ресурсів та програмного забезпечення

При створенні апаратної частини голосової аутентифікації, використовується апаратна платформа Arduino Uno, яка зображена на рисунку 2.1. До неї будуть приєднуватися усі інші компоненти.

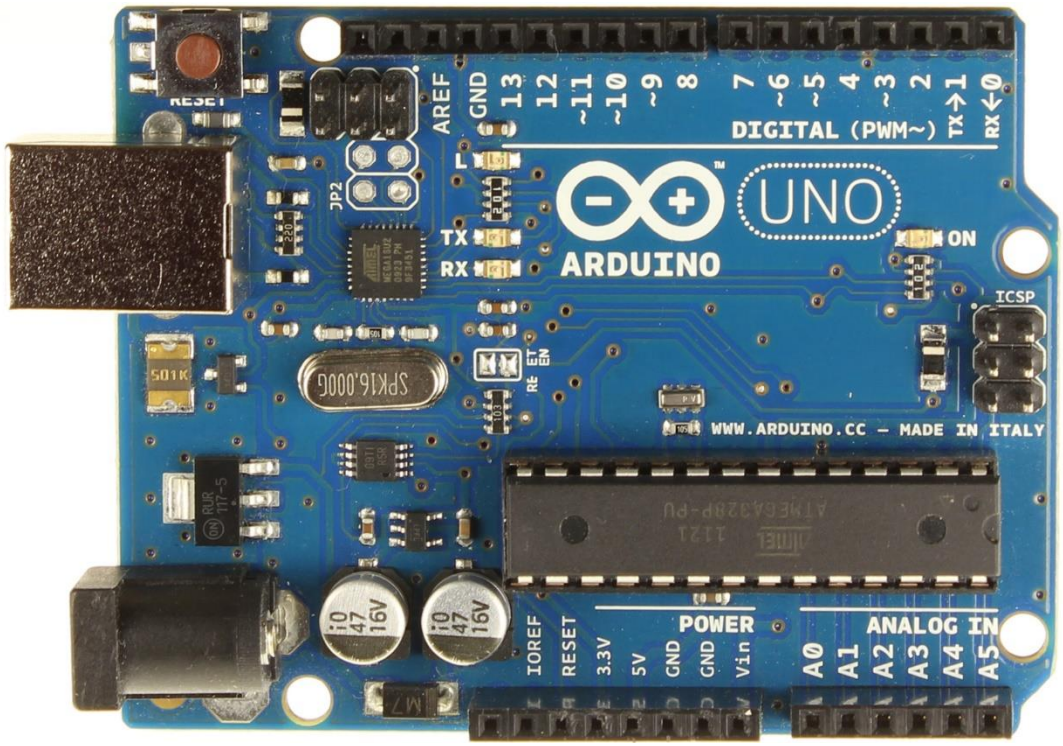


Рисунок 2.1 – Arduino Uno

Arduino UNO базується на мікроконтролері Atmega328P, розробленому Arduino.cc, що є платою мікроконтролера з відкритим вихідним кодом. Ця плата найбільш поширена, ніж інші плати Arduino.

Плата містить набори цифрових і аналогових контактів вводу/виводу (вхід/вихід), які можуть бути підключені до різних плат розширення (щитів) та інших схем [17-18].

Плата має 14 цифрових контактів, які можна використовувати як вхід/вихід. З цих 14 цифрових контактів 6 контактів можна використовувати як виходи ШІМ. D3, D5, D6, D9, D10 і D11 є контактами ШІМ в Arduino UNO.

ШІМ – це метод отримання аналогових результатів з цифрових засобів. ШІМ в основному використовується в Arduino для керування сервомоторами, світлодіодами, регулюванням швидкості тощо.

Плата має 6 аналогових контактів. Він має кварцовий кварцовий генератор на 16 МГц, який забезпечує тактовий сигнал мікроконтролеру Atmega 328 [19].

Плата має порт USB, який використовується для програмування плати, а також для живлення. Також він має роз'єм живлення, роз'єм ICSP, кнопку скидання та світлодіоди.

Також потрібен модуль на який буде завантажено потрібні голоси. Зручний при такому випадку модуль Voice Recognition, який зображений на рисунку 2.3.

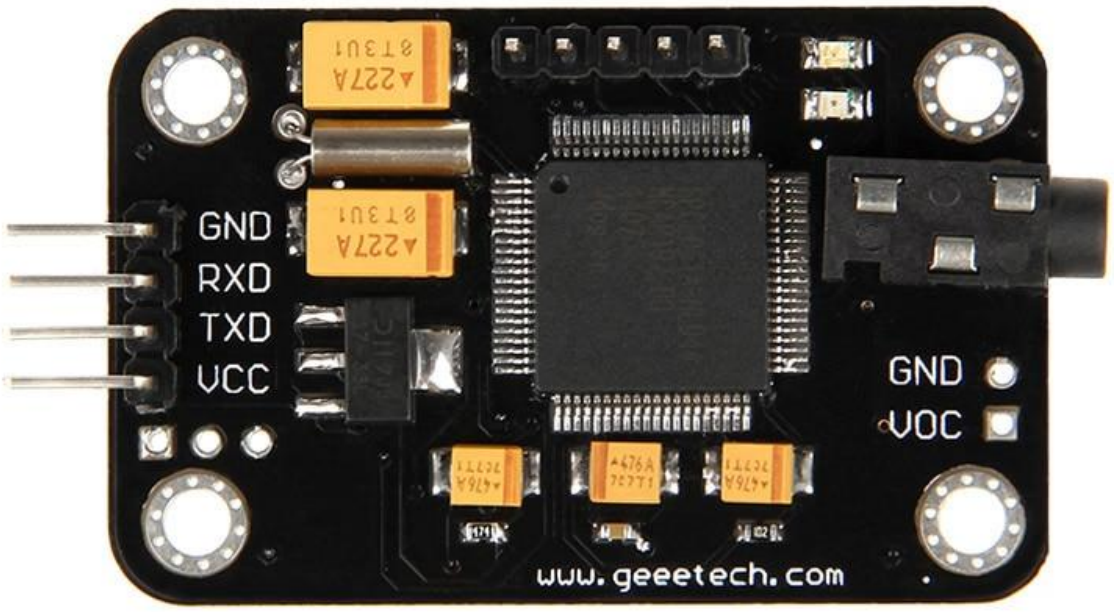


Рисунок 2.3 - Voice Recognition Module

Це компактний і простий в управлінні модуль розпізнавання мови. На основі цього модуля можна створювати проекти з голосовим керуванням.

Модуль має два види пам'яті: пам'ять сховища (де зберігається записана інформація) і пам'ять розпізнавача (де записана інформація бере участь в порівнянні з голосом, який надійшов з мікрофону) [20-21].

Перед розпізнаванням голосу, потрібно завантажити з пам'яті сховища в пам'ять розпізнавача. Пам'ять розпізнавача розрахована на 7 голосів, значить модуль здатний одночасно порівнювати до 7 голосів з звуковим сигналом, що надходить.

Така організація пам'яті дозволяє розділити довгі голосові команди (вимова яких займає більше 1,5 сек.) на дві і більш маленькі голосові команди, які будуть підвантажуватися зі сховища в розпізнавач у міру впізнання модулем попередніх частин довгої голосової команди.

До нього під’єднується мікрофон з роз'ємом jack 3,5 мм, який зображений на рисунку 2.4.

Рисунок 2.4 – Мікрофон з роз'ємом jack 3,5 мм

Завдяки ньому буде записуватися інформація та голос, яку мікрофон буде пропускати через себе та передавати на модулі і плати. Він гнучкий, легкий та чудова якість звуку.

Для того, щоб об’єднати усі деталі, потрібні кабелі jump wires. Зображені вони на рисунку 2.5 і 2.6 відповідно [22].

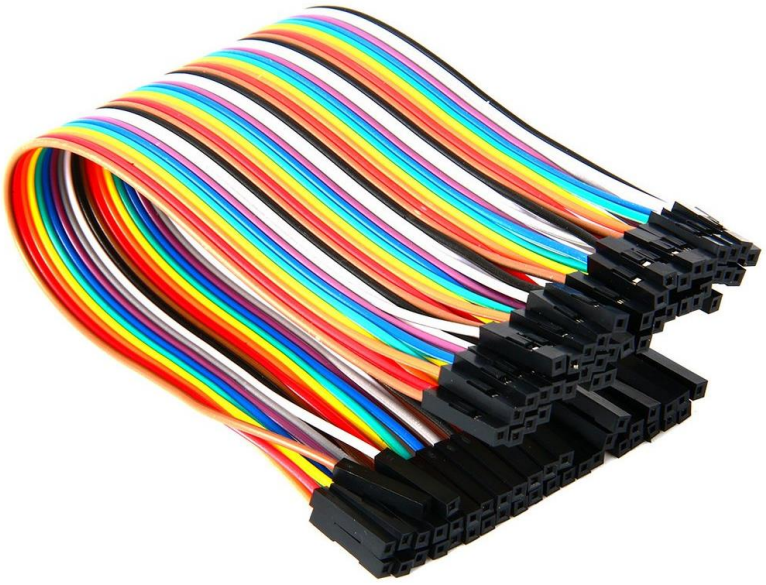


Рисунок 2.5 – Кабель jump wires f-f

Рисунок 2.6 – Кабель jump wires m-f

Кабелі jump wires - це просто дроти, які мають контакти на кожному кінці, що дозволяє використовувати їх для з'єднання двох точок один з одним без пайки. Ці кабелі бувають 3 типів: f-f (female-female), m-f (male-female) та m-m (male-male). Вони відрізняються тільки в кінцях кабелів. Male кінці мають штифт, що виступає і може вставлятися в речі, тоді як female кінці не мають і використовуються для підключення речей.

Так як у схемі буде дисплей, який буде показувати інформацію, що все пройшло добре і світлодіоди, нам потрібна макетна плата. Для схеми буде використана макетна плата Arduino MB-102, яка зображена на рисунку 2.7.

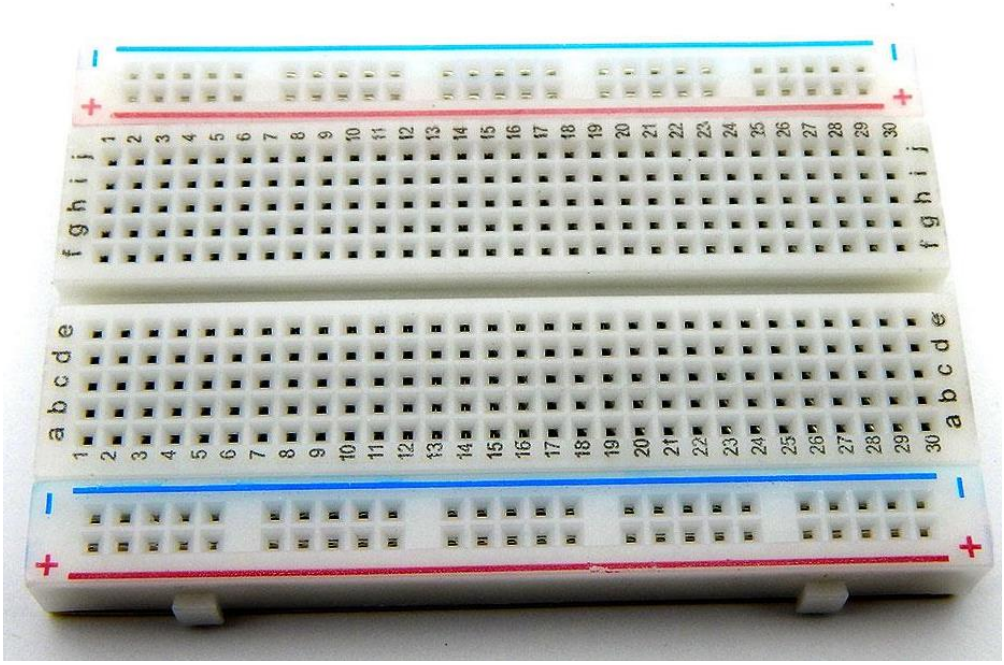


Рисунок 2.7 – Макетна плата Arduino MB-102

Макетна плата Arduino MB-102 для монтажу без пайки. Безпайкова плата використовується для Arduino проектів та інші стартапів в яких не потрібно проводити пайку елементів. На нижній частині плати є двосторонній скотч, що дозволить закріпити її на поверхні. Основною перевагою макетної плати Arduino це її простота і надійність кріплення радіодеталей без використання пайки. Всі елементи добре фіксуються в отворах макетної плати ардуїнів [23].

Щоб показувало кольорами, чи пройшла вірно аутентифікація, потрібно 2 світлодіоди: червоний (якщо не успішно пройшла аутентифікація) та зелений (якщо успішно пройшла аутентифікація). Вони зображені на рисунку 2.8.

Рисунок 2.8 – Світлодіоди

Також для того, щоб світлодіоди не перегоріли, потрібно встановити резистори, які зменшать опір на них, які зображені на рисунку 2.9.

Рисунок 2.9 – Резистор

Усі резистори діляться на лінійні та нелінійні. Опір лінійних резисторів не залежать від прикладеної напруги або струму, що протікає. Опір нелінійних резисторів змінюються в залежності від значення прикладеної напруги або струму, що протікає. Наприклад, опір освітлювальної лампи розжарювання за відсутності струму в 10-15 разів менше, ніж у режимі освітлення. У лінійних ланцюгах резистивних форма струму збігається з формою напруги, що викликав цей струм [24].

Також, щоб точно знати, чи успішно пройдено аутентифікація, потрібно встановити екран, який покаже відповідний текст. Для цього знадобиться LED екран 1602. Він показаний на рисунку 2.9.

Рисунок 2.9 – LED екран 1602

Символьний дисплей LCD1602 з блакитним підсвічуванням - рідкокристалічний дисплей (Liquid Crystal Display), екран якого здатний відображати одночасно до 32 символів (16 стовпців, 02 рядки). Підключення до Arduino здійснюється за синхронним 8-бітним паралельним інтерфейсом [25].

2.2 Вимоги до апаратного обладнання

Роблячи висновки з того, що було сказано в минулому пункті, є такі вимоги для створення системи голосової аутентифікації для розумного будинку:

1) апаратна платформа Arduino Uno;
2) модуль Voice Recognition;
3) мікрофон з роз'ємом jack 3,5 мм;
4) кабелі jump wires f-f та m-f;
5) макетна плата Arduino MB-102;
6) світлодіоди;
7) резистори;
8) LED екран 1602.

2.3 Вимоги до програмного забезпечення

Є середовище розробки Arduino, яке програмується за допомогою мов програмування C та C++, та є на всіх операційних системах (Windows, Linux, macOS). За допомогою цього середовища буде підключення плати, але написання коду буде за допомогою python [26-27].

Для того, щоб все працювало, спочатку в середовищі розробки Arduino підключаємо плату, а потім переходимо в будь яке середовище для написання коду на python. Потрібно встановити декілька бібліотек, які є в мові програмування, які зроблять роботу легшою. Це такі як json (для загрузки результатів), pyaudio (для взаємодії з мікрофоном), pyfirmata (для Arduino).

2.4 Вартість проекту

Можна зробити висновок, скільки часу та грошей потрібно витратити. У таблиці 2.1 буде показано, скільки буде коштувати пристрій.

Таблиця 2.1 – Вартість компонентів

Компоненти	Вартість
Arduino Uno	192 грн.
Voice Recognition	1372 грн.
Мікрофон	
Jump wires	295 грн.
Arduino MB-102	47 грн.
Світлодіоди	5 грн.
Резистори	3 грн.
LED екран 1602	98 грн.

У висновку, щоб створити пристрій, потрібно витратити 2012 гривень та декілька днів, щоб зібрати пристрій та створити програму. Також, десь потрібно 1 день, для того, щоб все підключити до розумного будинку.

2.5 Висновки

У цьому розділі було обґрунтовано такі моменти, як складові апаратного забезпечення для того, щоб створити систему голосової аутентифікації в розумному будинку. Було обговорено найкраще рішення для створення програмного забезпечення. Також, було обговорено ціну та час, який потрібно витратити, щоб створити проект та його установити в розумний будинок.

3 СТВОРЕННЯ ПРОГРАМНО-АПАРАТНОЇ ПІДСИСТЕМИ ТА ЇЇ ТЕСТУВАННЯ

3.1 Збірка програмно-апаратної частини

Для того, щоб зібрати схему, потрібно провести такі дії:

1) підключити LED екран до платформи Arduino Uno;
2) підключити модуль Voice Recognition до плати та платформи Arduino Uno;
3) підключити світлодіод та резистори до плати;
4) підключити мікрофон до модуля Voice Recognition.

На рисунку 3.1 зображена схема апаратної частини, яка зроблена у середовищі Fritzing.

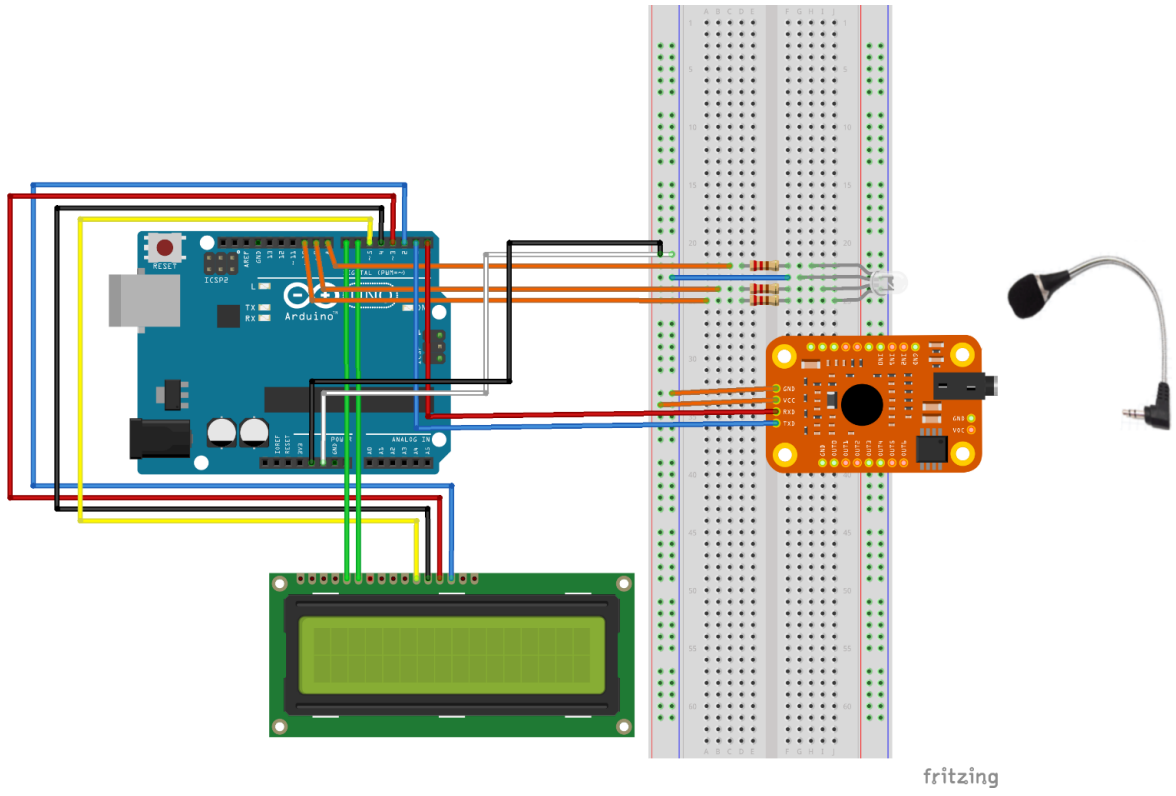


Рисунок 3.1 – Схема апаратної частини та її з’єднань

Спочатку було під’єднання LED екрану до Arduino Uno. Піни 5,6, 11, 12, 13, 14 з LED екрану були під’єднані до пінів 7, 6, 5, 4, 3, 2 плати Arduino Uno

відповідно. Потім підключено було модуль Voice Recognition до макетної плати за допомогою виходів GND і VCC і в подальшому підключені до Arduino Uno в SV і GND. Також виходи TXD і RXD відповідно були підключені до TXD і RXD до платформи. Потім до макетної плати було під'єднано світлодіод для відповідного кольору та резистор від перегорання. І на кінець до модуля Voice Recognition було під'єднано мікрофон. Тепер можна під'єднувати до ноутбука та все запускати.

3.2 Принцип роботи додатку для проходження голосової аутентифікації

Є багато циклів роботи, які відповідають за значення параметрів. Якщо треба переглянути залежність ефективності роботи голосової аутентифікації від тривалості фази говоріння людини, то треба, щоб система переглянула на різних тривалостях. До прикладу можна взяти тривалість від 0,6 секунд до 6 секунд з кроком 0,6 секунд. І кожному параметру тривалості буде відповідати свій цикл [28-29].

Цикл можна розбити на 2 фази: тренування зразками голосів осіб та проходження аутентифікації.

Також, можна кожну фазу розділити на ще декілька етапів. Для початку потрібно розказати про першу фазу – тренування зразками.

Ця фаза розбивається на такі етапи:

- розбиття текстів на задані тривалістю фази;
- вилучення голосових відбитків;
- тренування моделі по цим голосовим відбиткам.

Ці всі етапи можна побачити на відповідних блок-схемах, які розташовані нижче.

На рисунку 3.2 зображено блок-схему етапу розбиття тексту. Треба мати на увазі те, що заранння записані звукові тексти, які і подаються програмі для обробки.

Рисунок 3.2 – Блок-схема розбиття тексту

На рисунку 3.3 зображено схему вилучення голосових відбитків.

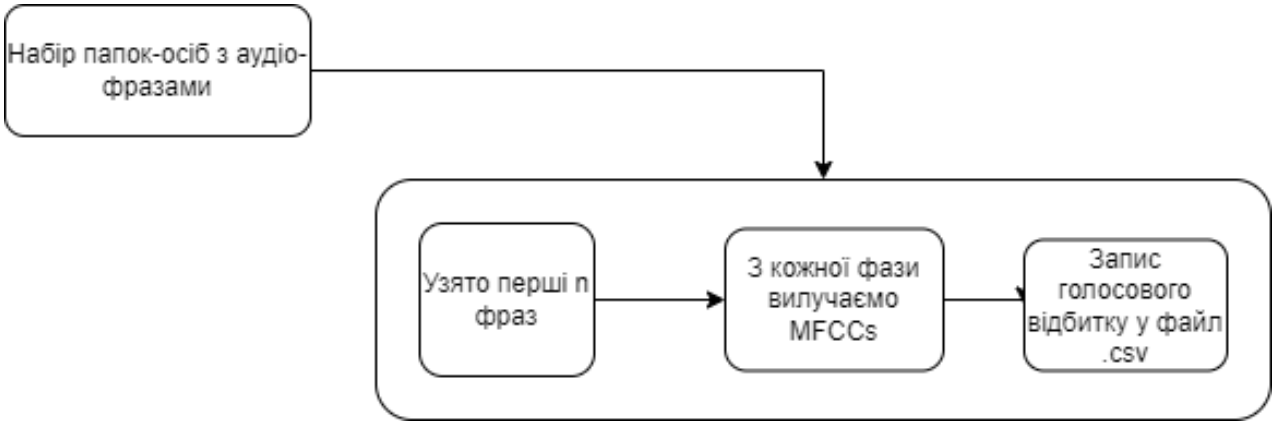


Рисунок 3.3 – Блок-схема вилучення голосових відбитків

А на рисунку 3.4 показано блок-схему етапу тренування моделі заданого типу.

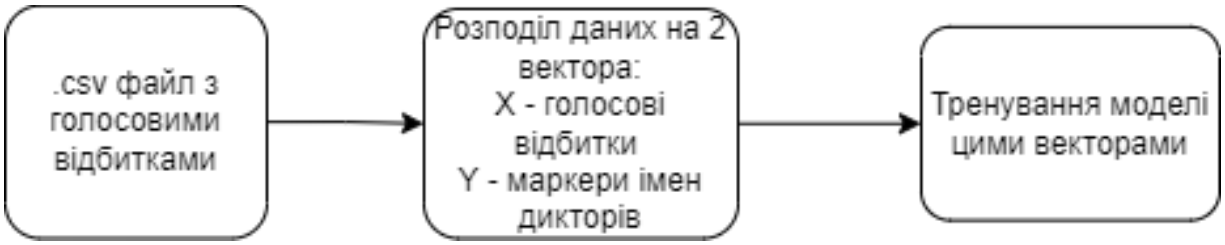


Рисунок 3.4 – Блок-схема тренування моделі

Також, треба розглянути і другу фазу – проходження аутентифікації. Вона також поділяється на етапи:

- вилучення голосового відбитку з фрази людини, яка проходить аутентифікацію;
- ідентифікація і перша верифікація людини;
- проходження другої верифікації за допомогою голосового відбитку та отримання остаточного результату.

Ці всі етапи можна побачити на відповідних блок-схемах, які розташовані нижче.

На рисунку 3.5 можна побачити блок-схему етапу вилучення голосового відбитку з фрази людини, яка проходить аутентифікацію [30].

Рисунок 3.5 – Блок-схема етапу вилучення голосового відбитку з фрази людини, яка проходить аутентифікацію

Блок-схему етапу ідентифікації і першої верифікації людини показана на рисунку 3.6.

А на рисунку 3.7 показано блок-схему етапу проходження другої верифікації за допомогою голосового відбитку та отримання остаточного результату.

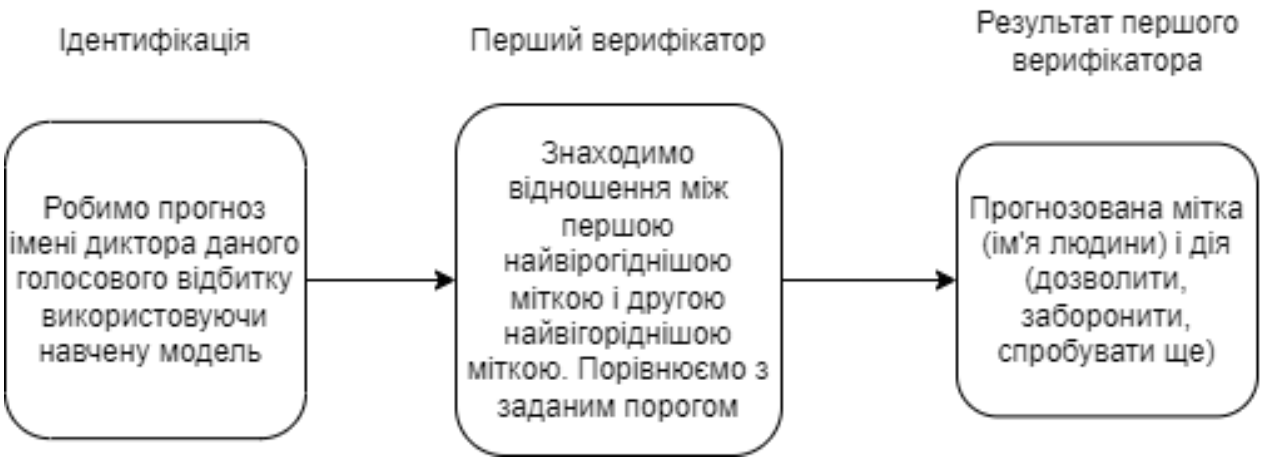


Рисунок 3.6 – Блок-схема етапу ідентифікації і першої верифікації людини

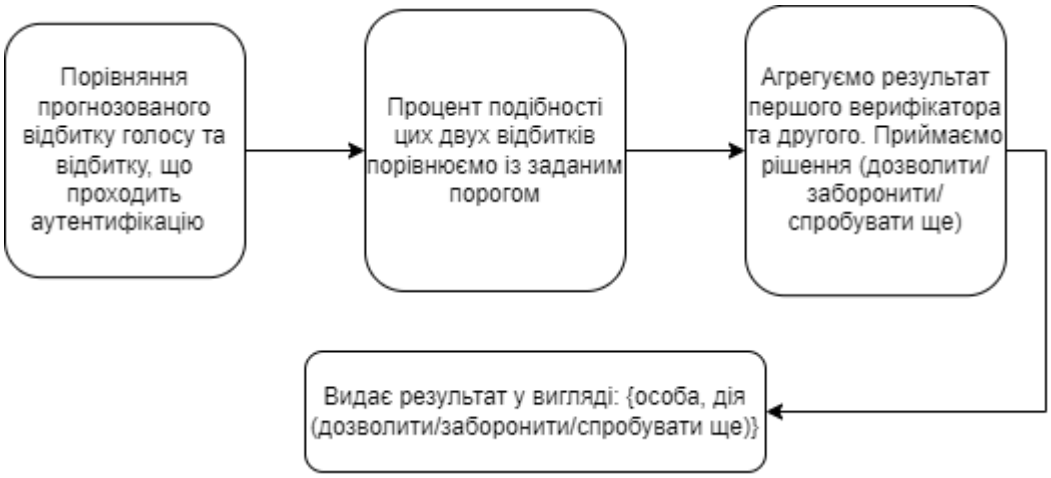


Рисунок 3.7 – Блок схема етапу проходження другої верифікації за допомогою голосового відбитку та отримання остаточного результату

А на рисунку 3.8 показано загальну схему додатку з голосовою біометрією для проходження голосової аутентифікації в розумному будинку.

Можна побачити на схемі, яка показана вище, що там є 2 модулі та модель.

Також, треба показати, як працюють ці модулі.

Спочатку, на рисунку 3.9 показано перший модуль, який відповідає за навчання системи голосом особи.

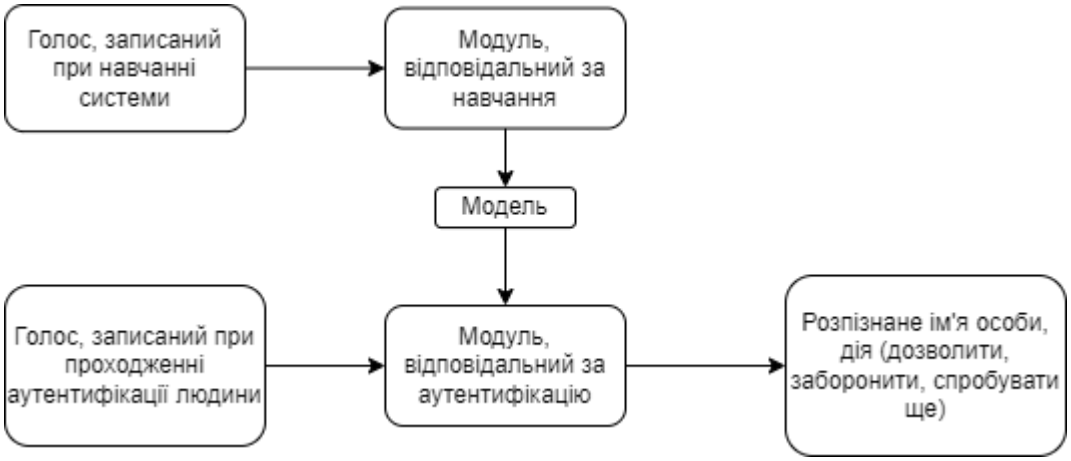


Рисунок 3.8 – Загальна схема додатку з голосовою біометрією для проходження голосової аутентифікації в розумному будинку

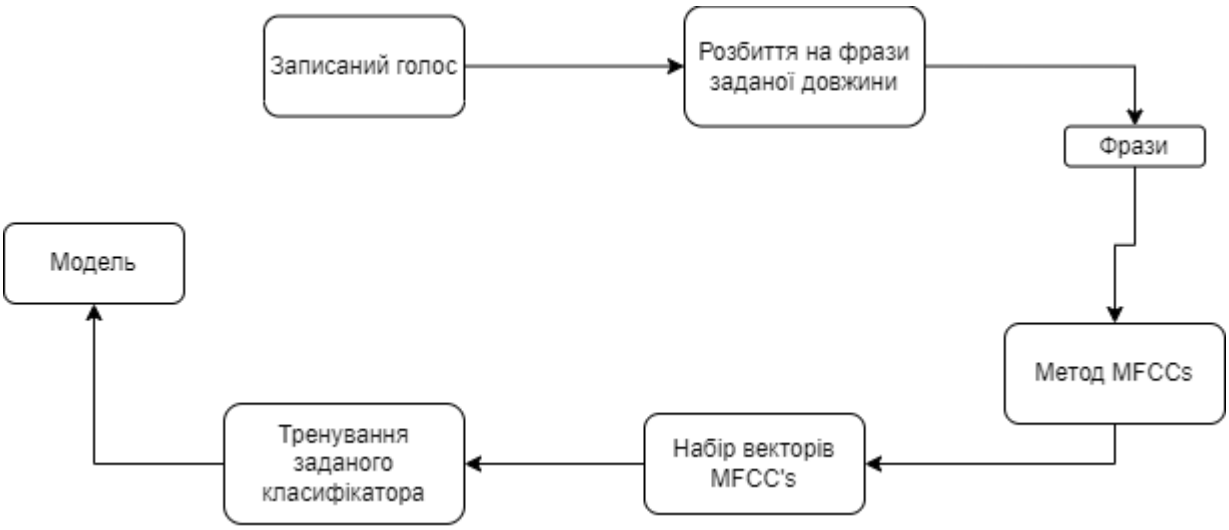


Рисунок 3.9 – Схема опису модуля за навчання системи

Можна побачити, що записаний голос спочатку розбивається на фази заданої довжини, що є дуже важливим фактором. Потім з вибирається голосовий відбиток. Також вдосконалені мають бути параметри нижньої та верхньої межі діапазону частот та FFT перетворення. А потім класифікатор вже займається обробкою голосового відбитку.

Завдяки цьому модулю отримаємо модель, яку можна використовувати на наступній фазі – аутентифікації людини (другий модуль). Його можна побачити на рисунку 3.10 [31].

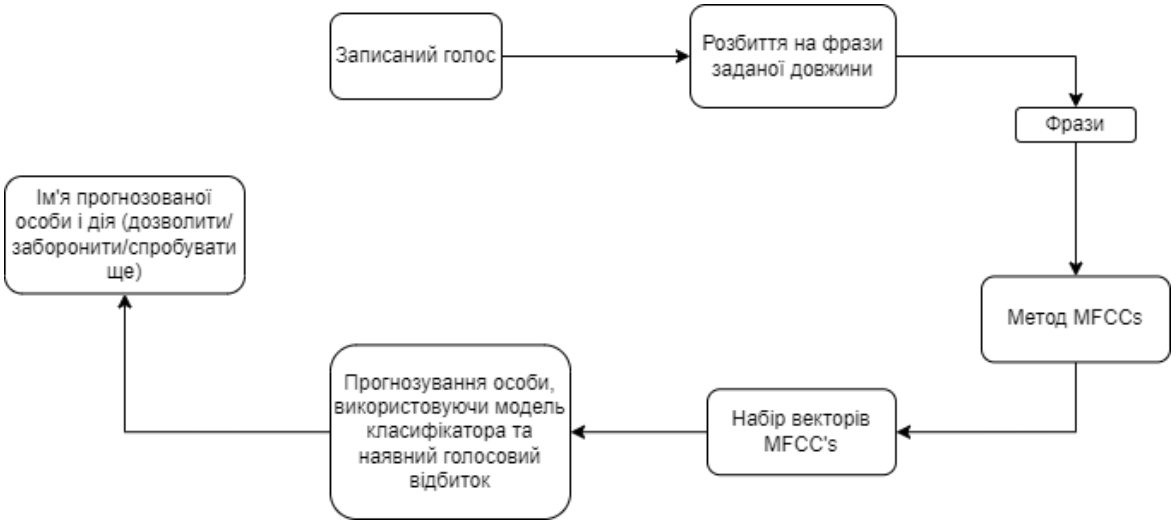


Рисунок 3.10 – Схема опису аутентифікації людини (робота першого верифікатора)

Тут все дещо схоже на попередню схему, але тут, завдяки тому, що створена модель за допомогою першого модуля можна прогнозувати ідентифікатор людини по її голосовому відбитку.

Також використовується відношення ймовірності прогнозу особи, що має найбільшу ймовірність розпізнавання до ймовірності другої. Можна зробити висновок, що чим більше це відношення, тим найбільша вірогідність того, що людина, що проходить аутентифікацію, пройде її успішно (це відношення буде мати назву, як firstMaxSecondMaxRelation).

Потрібно ще вдосконалити верхні і нижні пороги для параметра firstMaxSecondMaxRelation.

Можна зробити висновок, що якщо нижній поріг буде більшим, ніж firstMaxSecondMaxRelation, то потрібно заборонити вхід у дім, тому що особа, яка записана в базу та яка проходить аутентифікацію – це різні люди.

Якщо він буде більшим за верхній поріг, то потрібно дозволити вхід у дім, тому що цих два голоси ідентичні. Та якщо параметр firstMaxSecondMaxRelation буде більшим чи дорівнювати меншому порога та меншим за верхній поріг, то спробувати ще, тому що недостатньо зібраної інформації.

На рисунку 3.11 можна побачити схему опису другої частини модуля голосової аутентифікації.

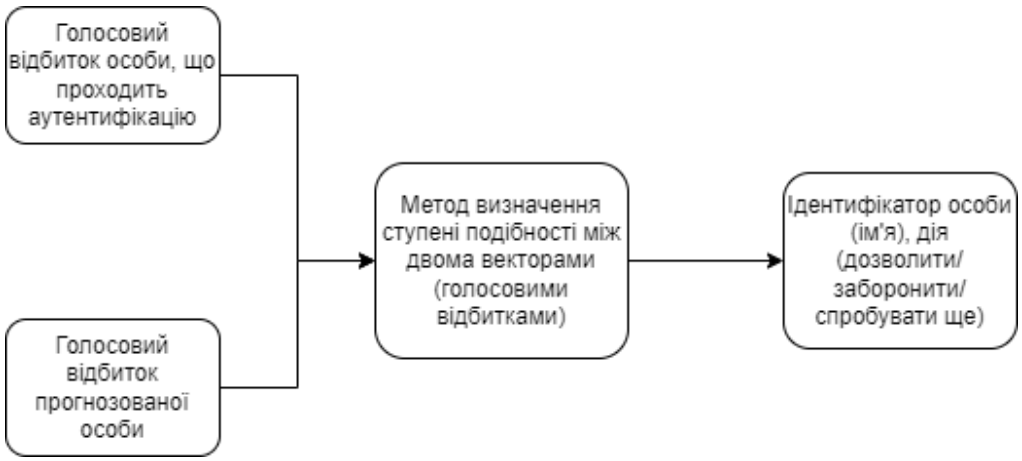


Рисунок 3.11 – Схема опису другого верифікатора модуля голосової аутентифікації

Тут уже є другий ступінь верифікації. А точніше перевірка подібності голосового відбитку людини, яка йде на порівняння з тим, яка є в базі.

Для цього використовується метод One-class SVM. Він має показати, що 2 вектори однакові.

Для того, щоб показати, що вектори два однакові, треба створити ще одне співвідношення. З цього можна зробити висновок, що завдяки цьому співвідношенню можна вирішити, чи дати дозвіл, заборонити чи спробувати ще.

Візьмемо так, що співвідношення матиме назву comparisonRelation. І тому, якщо співвідношення comparisonRelation менше за нижній поріг, то потрібно заборонити дозвіл, тому що це різні люди.

Якщо більше ніж верхній – дозволити, тому що це ідентичні люди, та якщо більше ніж менший поріг, та нижче ніж верхній – спробувати ще.

На рисунку 3.12 зображено схему опису останньої частини роботи аутентифікації особи.

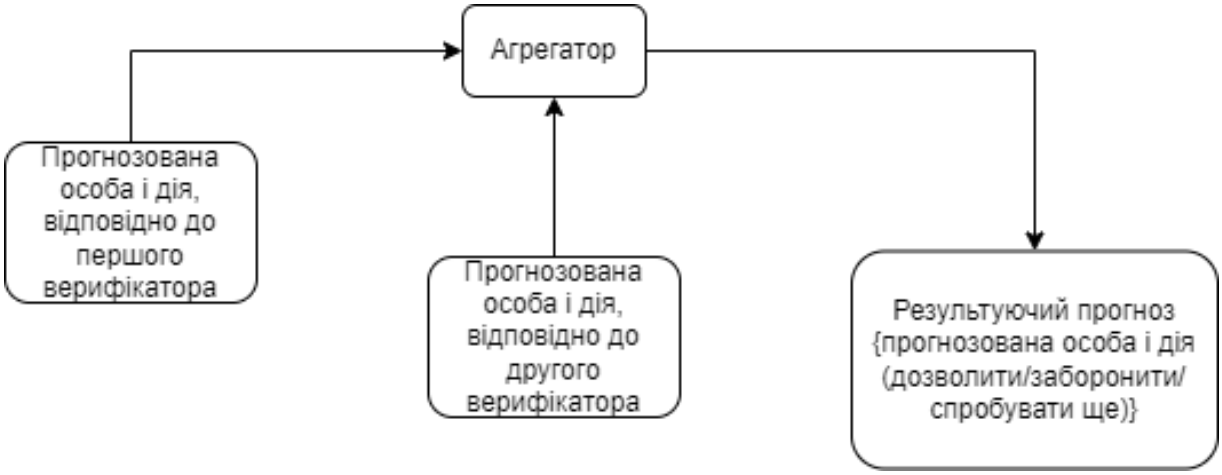


Рисунок 3.12 – Опис схеми агрегації двох верифікаторів для голосової аутентифікації

Перший та другий верифікатор позначимо, як IdentRelation VerifyRelation, а агрегатор - Result, то це все має таку логіку в коді:

```
If (IdentRelation == “Дозволити”):
    Result = “Дозволити”
  else:
    if(IdentRelation == “Заборонити”):
      Result = “Заборонити”
    else:
     IdentRelation = “Спробувати ще”
      if VerifyRelation == “ Дозволити”:
        Result = “Дозволити”
      else:
        if VerifyRelation == “Заборонити”:
          Result = “ Заборонити”
    IdentRelation == “ Спробувати ще” and VerifyRelation == “ Спробувати ще”
        else:
            Result = “ Спробувати ще”
```

Після усього цього буде виведений результат: ім’я людини та дія – дозволити доступ до дому, заборонити чи запросити записати голос ще раз.

3.3 Додаток

Ось так виглядає програма, де оброблюється, фільтрується та аналізує звуковий сигнал (рисунок 3.13).

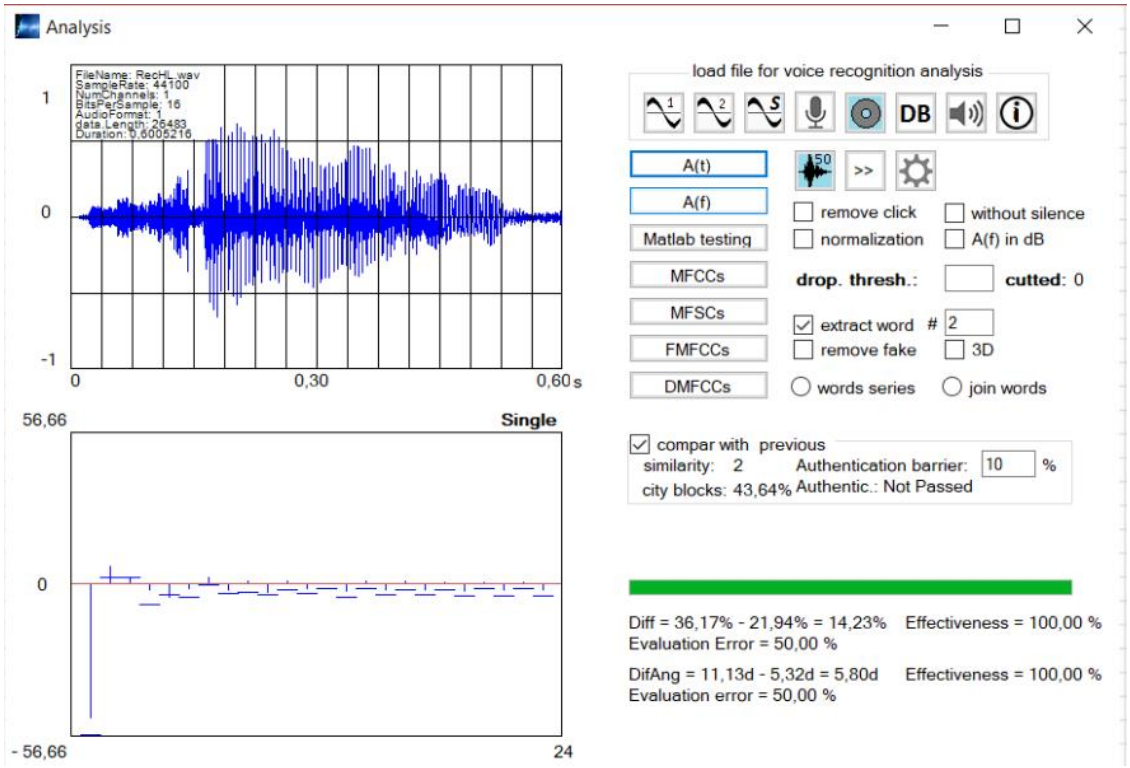


Рисунок 3.13 – Вікно програми

На рисунку можна побачити, що у верхньому лівому кутку є графік, на якому можна побачити, як обробляється звук.

У нижньому лікому кутку можна побачити графік, на якому показується графіки n елементів векторів енергетичного розподілу E спектру голосових сигналів та векторів-ознак C, складених з n мел-кепстральних коефіцієнтів шаблонного та аналізованого голосових сигналів.

Також внизу є лінія, яка показує прогрес обробки, налаштування, сторінка з поясненням, як користуватися програмою.

Також там є функції, де можна переглянути, як обробляються MFCC та інші функції.

Далі показано те, як виглядає оброблення звуку поетапно (рисунок 3.14).

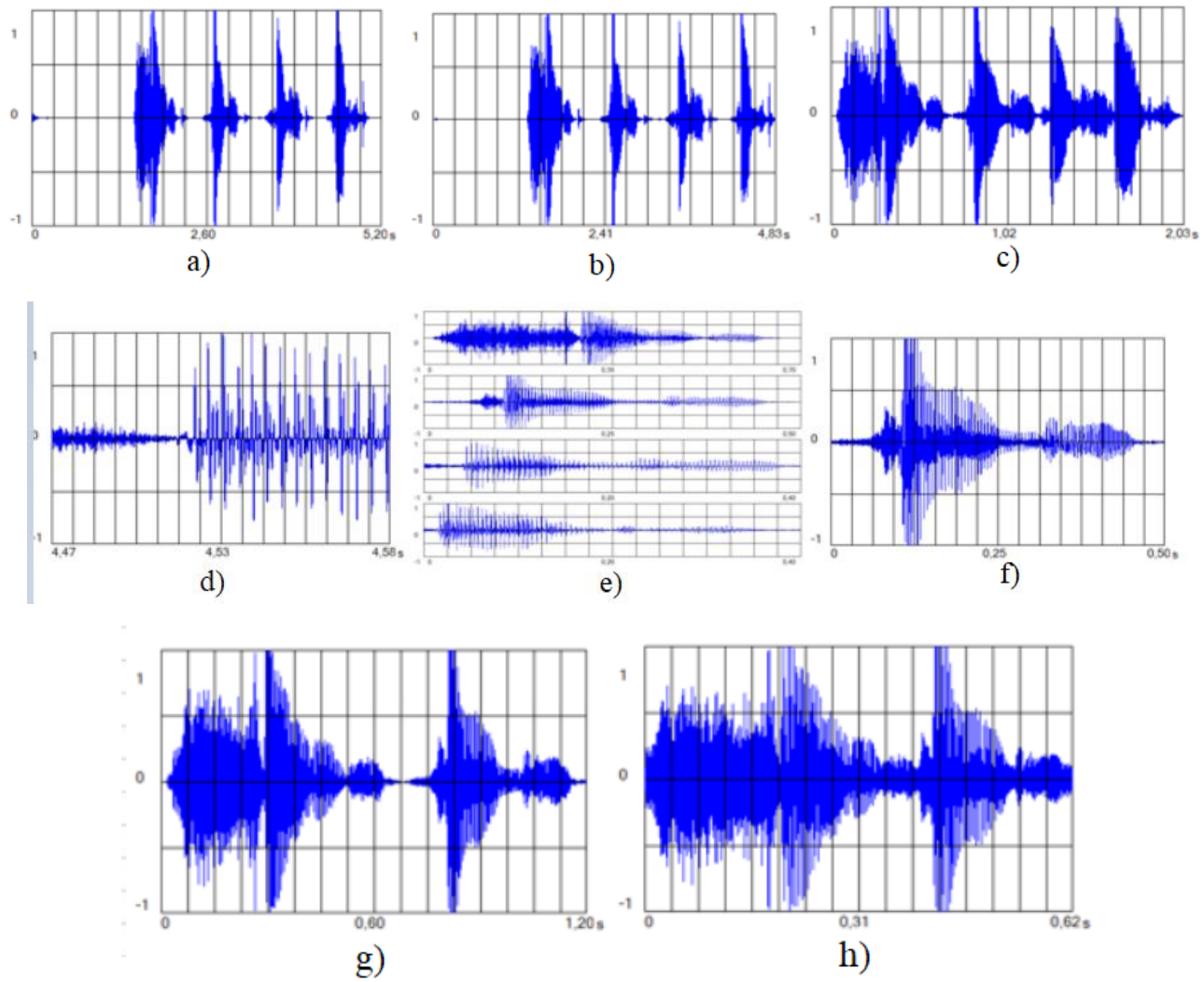


Рисунок 3.14 – Обробка звуку

Ось так виглядає обробка звуку.

В пункті a) йде нормалізація, в b) і c) - очистка від «кліків», в d) - видалення непотрібної тиші в сигналі, в e) – розбивка фрази на слова, f) – виймання особливостей голосу, g) - видалення сильно зашумлених ділянок і з’єднання тих, що залишились в один образ, та h) - порівняння голоса з правильним зразком.

На рисунку 3.15 показано вікно, де записується користувач, який буде вірний для входу та вікно, де показує результат.

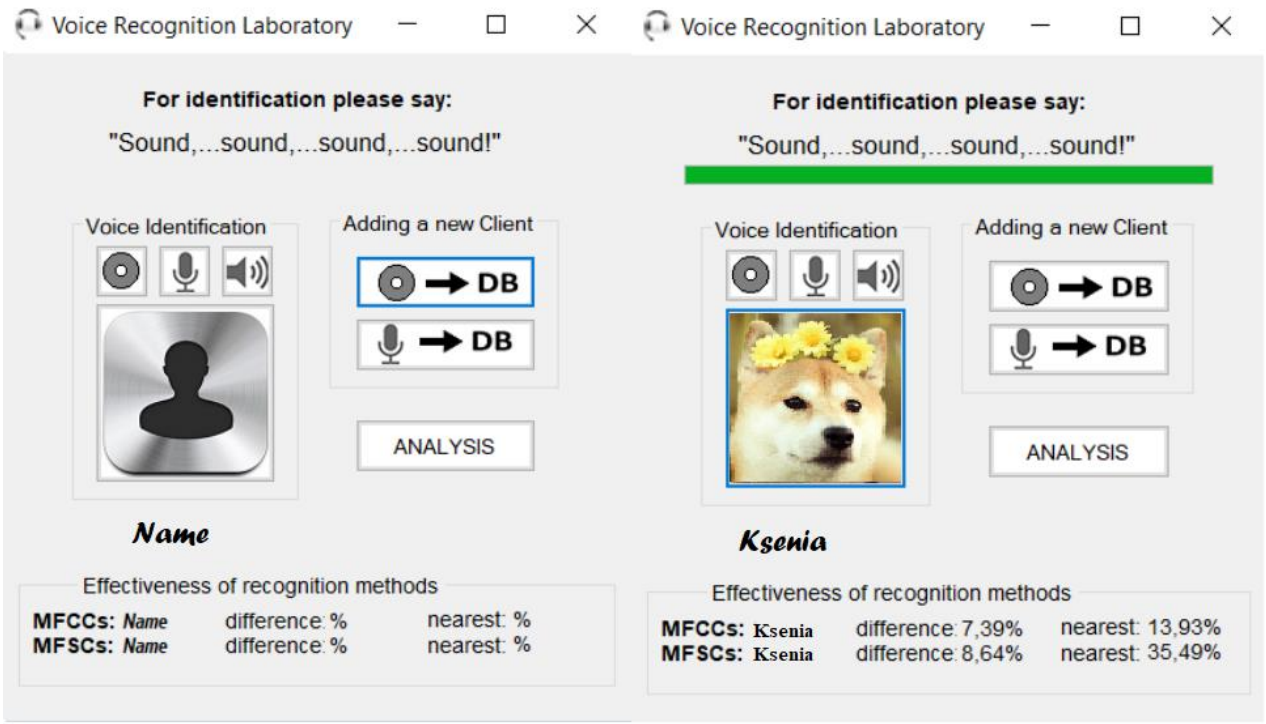


Рисунок 3.15 – Вікно запису та результат

Також можна використати програму на телефоні, яка буде основана на HMM. Вона виводить мел-кепстральні коефіцієнти та показує результат: дозволити чи заборонити вхід. Це все потрібно для того, що якщо щось випадково стане з пристроєм і була заміна йому.

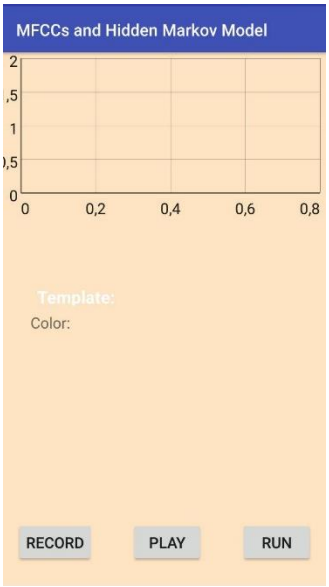
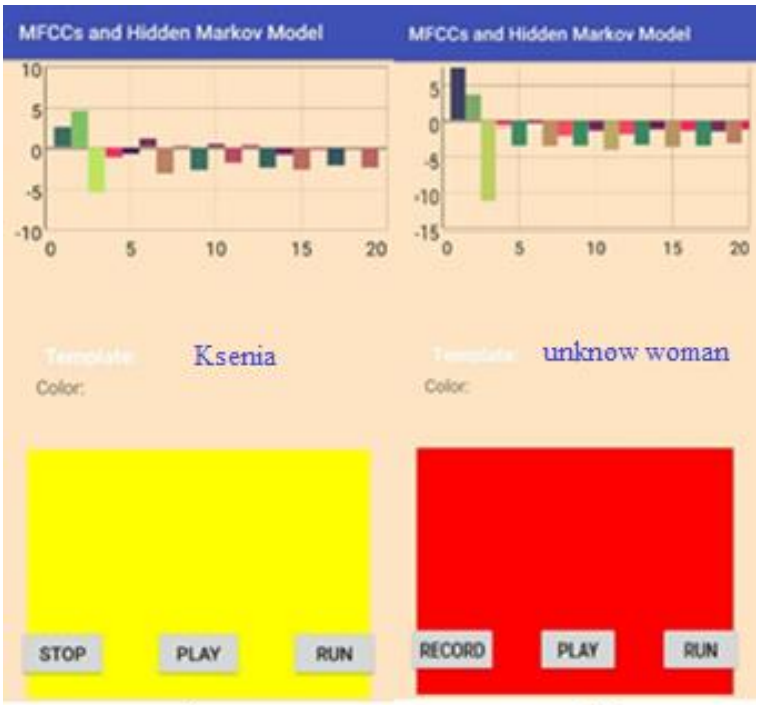


Рисунок 3.16 – Скріншот програми

На рисунку 3.16 показано інтерфейс даної програми. Можна побачити кнопку Record, за допомогою якої, людина записує голос, Run, яка запустить перевірку, та Play, яка покаже результат.

На рисунку 3.17 показано успішне (жовтим кольором) та не успішне (червоним кольором) проходження ідентифікації.



а) б)

Рисунок 3.17 – а) успішне проходження, б) не успішне проходження

3.4 Тестування приладу

Спробуємо записати власний голос. Щоб записати голос в програму, потрібно відкрити файл writeVoice.py. Коли програма попросить ввести ім’я, то вводимо і натискаємо на клавішу «Enter». Після чого з’являється текст, який треба читати до тих пір, поки не відобразиться текст «Фраза записана». На рисунках 3.18 та 3.19 зображено консоль під час надиктовки тексту та коли воно записало.

Після того, як було записано достатню кількість голосів, можна перейти на фазу перевірки роботи аутентифікації. Треба запустити файл authentication.py і

прочитати текст, який буде відображатись. Приклад тексту зображений на рисунку 3.20. Система зробить запис перших 6 секунд. Програма зупинить запис, як людина зачитає 6 секунд тексту і відобразить надпис «Фраза записана» (Allow – дозволити, Deny – заборонити, Try Again – спробувати ще).

```
введіть своє ім"я (на англійській): ksenia
================================================================================
прочитайте фразу в мікрофон поки не вісвітиться повідомлення (20 сек):
================================================================================
мисливців-збирачів. Археологи знайшли близько 800 пам'яток цих людей в Україні і виділяють їх у закарпатську, дністровську
, волинську,
середньодніпровську та степову групи.[8] Серед них особливо виділяють природний останець Кам'яна Могила, який став культов
им центром
кроманьйонців степової зони[9].

Понад 10 тисяч років тому відбувся перехід від палеоліту до мезоліту, який збігся з таненням льодовика та початком нової г
еологічної
доби — голоцену. Загальне потепління сприяло збільшенню кількості населення[10]. Проте криза привласнюваного мезолітичного
 господарства
поступово змусила людей приступити до відтворювальних форм: рільництва і скотарства. Це сприяло винаходу кераміки. Настала
 нова доба
неоліту, яка тривала протягом 6—4 тисячоліть до н. е. Стабілізувався ландшафтний поділ України на лісову, лісостепову і ст
епову зони,
утворився гумусний покрив землі. Неолітичні культури України формувалися під впливом досягнень осередків Близького Сходу,
які імпортувалися
переважно через Балканський півострів і Подунав'я[11].
```

Рисунок 3.18 – Консоль програми під час надиктовки тексту для запису в програму

```
================================================================================
================================================================================
================================================================================
================================================================================
================================ФРАЗА ЗАПИСАНА!================================
================================================================================
================================================================================
================================================================================
================================================================================
mfccs було записано в .csv
```

Рисунок 3.19 – Консоль програми після надиктовки тексту для запису в програму

```
================================================================================
прочитайте фразу в мікрофон поки не вісвітиться повідомлення (6 сек):
================================================================================
Пантікапей (біля сучасної Керчі), Феодосія. Ці держави були демократичними або аристократичними за устроєм. У цих містах
панували рабовласницькі відносини. Головним джерелом постачання рабів був військовий полон, народження від рабині чи купів
ля
на невільничих ринках. Права у полісі мали лише повнолітні чоловіки-греки, уродженці міста. Колонія складалася власне з по
ліса
та сільськогосподарських округів. Міста оснащено водогоном та водостоком, були поширені ремесла й торгівля. Міста карбувал
и власні монети.

Боспорська держава (V ст. до н. е. — IV ст. н. е.) займала територію сучасного Керченського та Таманського півострова.
До складу царства увійшли такі міста, як Феодосія, Фанагорія, а столицею був Пантікапей. Спершу це був союз полісів, які
мали певну автономію, та згодом це об'єднання перетворилося на абсолютну монархію. Економіка цього царства була побудована

на сільському господарстві та торгівлі з Афінами, куди вивозили до 5 млн. пудів зерна. У I ст. до н. е. відбулося об'єднан
ня
під владою понтійського царя Мітрідата VI більшості Північного Причорномор'я, але він зазнав поразки від римлян.
```

Рисунок 3.20 – Консоль програми під час надиктовки тексту для аутентифікації

На рисунку 3.21 показано успішне проходження аутентифікації.

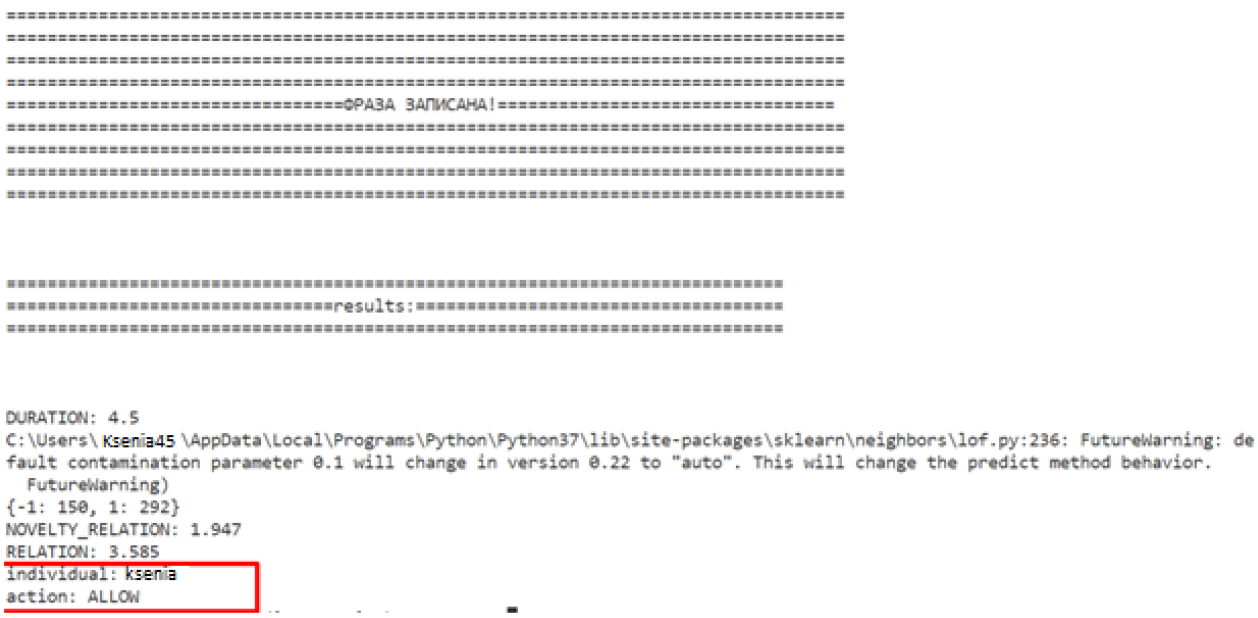


Рисунок 3.21 – Консоль програми після надиктовки тексту для аутентифікації

На рахунок швидкості опрацювання програми. Середня тривалість обробки голосу займає 24 секунди, з них 20 секунд запис голосу. А при перевірці аутентифікації 6 секунд був запис голосу та 2 секунди порівняння та виведення результату.

Точність додатку також була проведена. Було обрано експеримент з 30-ма голосами. 20 з них було записано в систему, а перевірено но 30 голосів. Це було зроблено для того, щоб перевірити систему на стійкість до злочинців. Тобто, 10 з 30 не були записані в систему і вони були як голоси злочинців.

У таблиці 3.1 показано результат цього експерименту, де relation – відношення ймовірності людини, яка має найбільшу ймовірність до ймовірності другої за рахунком особи з найбільшою ймовірністю. Novelty_relation – показник дії по перевірці двох параметрів голосів. Action – дія, яку пропонує програма (як описувалось вище Allow – дозволити, Deny – заборонити, Try Again – спробувати ще). Status – результат (true – прогноз пройшов успішно, false – прогноз пройшов не успішно).

Таблиця 3.1 – Результат експерименту

№	Relation	Novelty_relation	Action	Status
1	7,7	0,46	Allow	true
2	4,1	0,56	Allow	true
3	12,28	0,74	Allow	true
4	3,77	0,6	Allow	true
5	5	0,54	Allow	true
6	6,81	0,66	Allow	true
7	8,22	0,57	Allow	true
8	7,09	0,64	Allow	true
9	24,95	0,6	Allow	true
10	3,21	0,8	Allow	true
11	22,93	0,69	Allow	true
12	2	0,71	Allow	true
13	2,78	0,7	Allow	true
14	10,04	0,73	Allow	true
15	4,58	0,76	Allow	true
16	3,48	0,53	Allow	true
17	11,42	0,56	Allow	true
18	10,92	0,61	Allow	true
19	439	0,92	Allow	true
20	1,74	0,38	Try Again	
21	1,31	0,24	Try Again	
22	1,15	0,48	Deny	
23	1,12	0,29	Deny	
24	1,73	0,58	Allow	false
25	1,32	0,22	Try Again	
26	1,49	0,07	Deny	

Кінець таблиці 3.1 – Результат експерименту

27	1,17	0,21	Deny	
28	1,95	0,35	Try Again	
29	1,01	0,06	Deny	
30	1,24	0,17	Deny	

Завдяки цій таблиці можна підрахувати статистику:

- кількість дозволів та успішних прогнозів – 19;
- кількість дозволів та не успішних прогнозів – 1;
- кількість запропонувань спробувати ще – 4.

Завдяки цій таблиці, можна зрозуміти, які потрібні параметри порогів, щоб були вони задовільні.

Для параметру firstMaxSecondMaxRelation верхній поріг буде 5, а нижній десь між 1,2-2. А для параметру comparisonRelation верхній поріг буде 0,2, а нижній – 0,4.

Потрібно зробити декілька перевірок: перевірити систему при усіх значеннях FFT перетворенні і тривалості, де нижній поріг буде 1,2 та при 2 на 3 особах.

У таблиці 3.2 показано першу перевірку з нижнім порогом 1,2, де FFT перетворення має значення від 256 до 65525, а тривалість фрази від 0,6 до 6. До знаку / показує кількість правильних розпізнавань голосів, і чим більше число, тим краще, а після показує кількість помилок, що чим менше число, тим краще.

Ця перевірка виконувалась з злодіями, чий голос не записаний був у систему (75% голосів були записані), та без них.

Таблиця 3.2 – Результат системи з нижнім порогом 1,2

FFT/Dur	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	
256	29/29 47/17	32/32 57/5	44/22 70/0	47/23 71/0	48/23 71/0	48/20 71/0	48/23 71/0	48/15 71/0	48/21 70/0	48/23 71/0	With thief Without thief
512	23/13 31/2	23/16 38/2	42/7 61/0	45/8 68/0	48/14 70/0	48/8 68/0	48/21 67/0	48/1 67/0	48/4 70/0	48/11 71/0	With thief Without thief
512	29/15 46/10	35/23 60/7	46/19 69/0	48/20 71/0	48/22 71/0	48/22 71/0	48/23 71/0	48/19 71/0	48/22 71/0	48/22 71/0	With thief Without thief
2048	26/13 33/1	28/10 45/0	40/2 60/0	48/7 71/0	48/7 70/0	48/13 71/0	48/7 71/0	48/2 71/0	48/5 71/0	47/1 71/0	With thief Without thief
1024	31/15 45/8	41/19 63/4	45/6 69/0	48/12 71/0	48/13 71/0	48/21 71/0	48/18 71/0	48/16 71/0	48/16 71/0	48/16 70/0	With thief Without thief
8192	27/18 32/4	27/10 37/0	41/1 62/0	44/2 66/0	48/2 69/0	46/3 70/0	48/2 71/0	48/0 71/0	47/1 71/0	47/0 70/0	With thief Without thief
2048	29/17 38/4	33/12 51/4	44/4 67/0	48/13 71/0	48/11 71/0	48/17 71/0	48/9 71/0	48/6 71/0	48/5 71/0	47/3 71/0	With thief Without thief
32768	27/17 30/4	27/10 35/0	41/2 62/0	44/2 66/0	48/1 69/0	46/3 70/0	48/2 71/0	47/0 70/0	47/1 71/0	48/0 71/0	With thief Without thief
4096	33/13 39/4	26/11 40/0	44/2 67/0	44/4 67/0	46/3 69/0	47/6 70/0	48/2 71/0	48/0 71/0	48/1 71/0	48/0 71/0	With thief Without thief

На рисунках 3.22 і 3.23 показано графіки вдалого проходження аутентифікації зі злодіями і без.

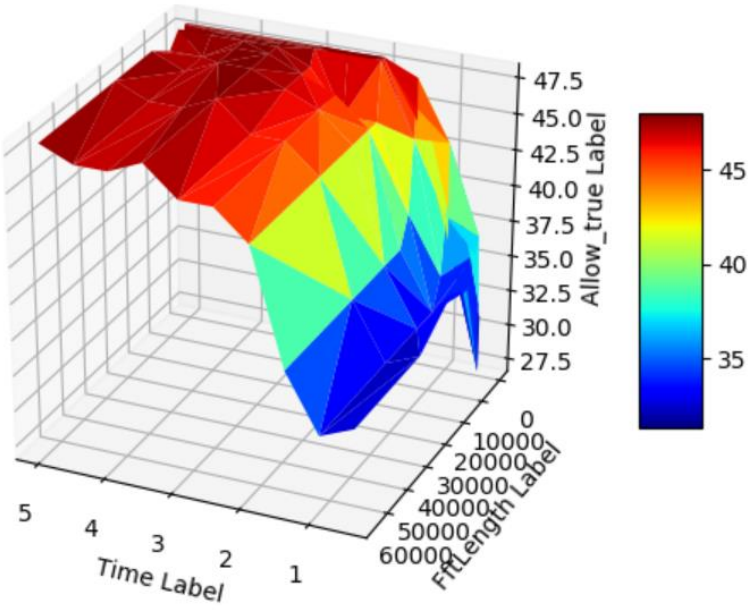


Рисунок 3.22 – Графік успішного проходження зі злодіями

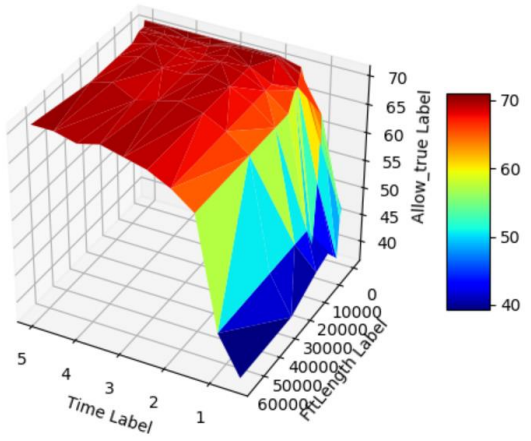


Рисунок 3.23 – Графік успішного проходження без злодія

На рисунках 3.24 і 3.25 показано графіки не вдалого проходження аутентифікації зі злодієм і без.

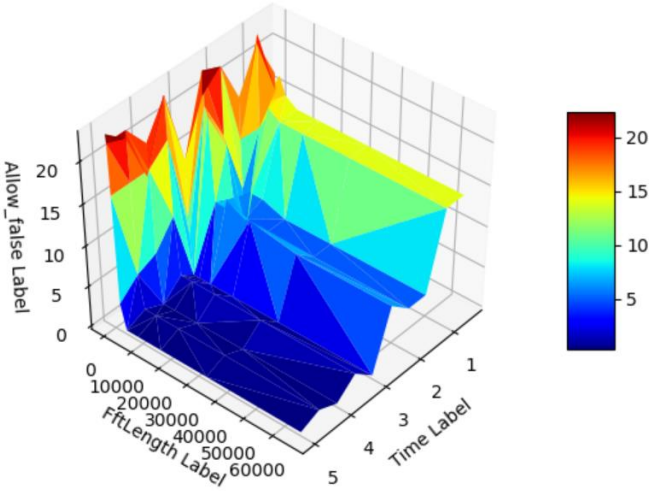


Рисунок 3.24 – Графік не успішного проходження з злодієм

Можна зробити висновок, дивлячись з таблиці, яка розміщена вгорі, що при збільшенні FFT перетворень та тривалості, збільшується і якість аутентифікації.

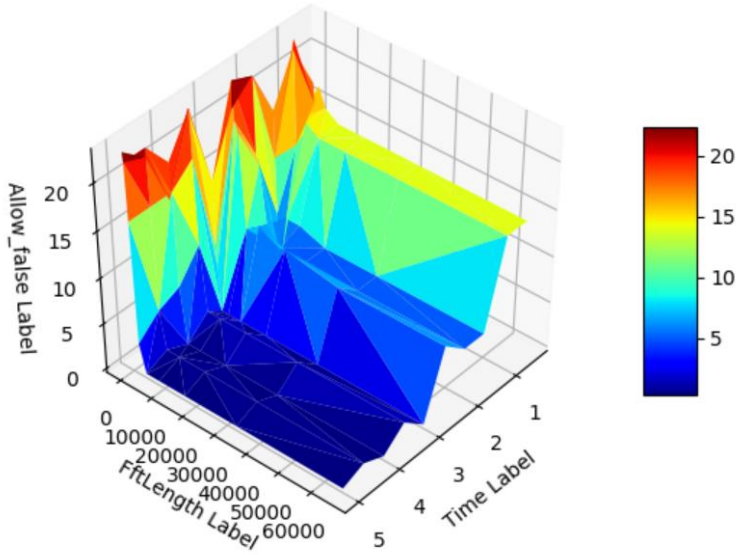


Рисунок 3.25 – Графік не успішного проходження без злодія

Також можна помітити, що для експерименту без злодія для FFT дуже важливе значення більше за 4096 і тривалість 1, тому що система не пропускає ніякого злодія.

Чудово те, що при збільшенні FFT зменшується помилки експерименту без злодія та з ним. Це відбувається, тому що при збільшенні FFT, росте і роздільна здатність системи, тому і росте ефективність аутентифікації.

Роблячи висновку з таблиці 3.2, то можна побачити, що максимальне значення правильного розпізнавання при експерименті з злодієм і без і мінімальним значенням помилок є тривалість, яка ≥ 4 і FFT ≥ 4096.

Найкращими параметрами для FFT є 8192, а для тривалості – 4,5 секунди.

Знову було проведено експеримент з 30 голосами, де 10 з них злодіїв (для одного значення FFT був об’єм 70 разів), який був показаний у таблиці 3.3.

Таблиця 3.3 – Результат експерименту, де нижній поріг = 1,2, а тривалість від 2 до 6 секунди.

	256	512	1024	2048	4096	8192	16384	32768	65536
2.0-6.0	48/7	49/5	46/3	43/1	42/1	42/1	43/1	43/1	43/1

Ще раз можна побачити те, що найкращий варіант FFT ≥ 2048, де пропустило тільки 1 злодія.

У таблиці 3.4 показано тепер друга перевірка з нижнім порогом 2, де FFT зі значеннями від 256 до 65525, а тривалість від 0,6 до 6.

Таблиця 3.4 – Результат системи з нижнім порогом 2

c	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	
256	27/18 35/4	27/10 37/2	41/1 57/0	44/2 64/0	48/2 66/0	46/3 59/0	48/2 63/0	48/0 62/0	47/1 64/0	47/0 70/0	With thief Without thief
512	23/13 31/2	23/16 38/2	42/7 61/0	45/8 68/0	48/14 70/0	48/8 68/0	48/21 67/0	48/1 67/0	48/4 70/0	48/11 71/0	With thief Without thief
1024	25/12 31/4	32/12 49/2	40/1 59/0	48/3 70/0	48/8 70/0	48/12 71/0	48/15 70/0	48/2 70/0	48/12 71/0	48/9 70/0	With thief Without thief
2048	26/13 33/1	28/10 45/0	40/2 60/0	48/7 71/0	48/7 70/0	48/13 71/0	48/7 71/0	48/2 71/0	48/5 71/0	47/1 71/0	With thief Without thief
4096	26/13 32/3	25/11 34/0	42/2 64/0	44/4 66/0	46/3 68/0	47/4 70/0	48/2 71/0	48/0 71/0	48/1 71/0	48/0 71/0	With thief Without thief
8192	27/18 32/4	27/10 37/0	41/1 62/0	44/2 66/0	48/2 69/0	46/3 70/0	48/2 71/0	48/0 71/0	47/1 71/0	47/0 70/0	With thief Without thief
16384	27/18 31/4	27/10 37/0	41/1 62/0	44/2 66/0	48/1 69/0	46/3 70/0	48/2 71/0	48/0 71/0	47/1 71/0	48/0 71/0	With thief Without thief
32768	27/17 30/4	27/10 35/0	41/2 62/0	44/2 66/0	48/1 69/0	46/3 70/0	48/2 71/0	47/0 70/0	47/1 71/0	48/0 71/0	With thief Without thief
65536	27/17 30/4	27/10 35/0	41/1 62/0	44/1 66/0	48/1 69/0	46/3 70/0	48/2 71/0	47/0 70/0	47/1 71/0	48/0 71/0	With thief Without thief

При порівнянні останніх 2 рисунків, можна помітити, що низькому порогу 2 результат хороший, а FFT в межах 256 до 4096. А коли він більше 8192, то результати майже однакові. Що можна зробити висновок, що найкращий варіант низького порогу є 2.

На рисунках 3.26 і 3.27 показано графіки успішного проходження з злодіями і без.

А на рисунках 3.28 і 3.29 показано графіки не успішних проходжень з злодієм і без.

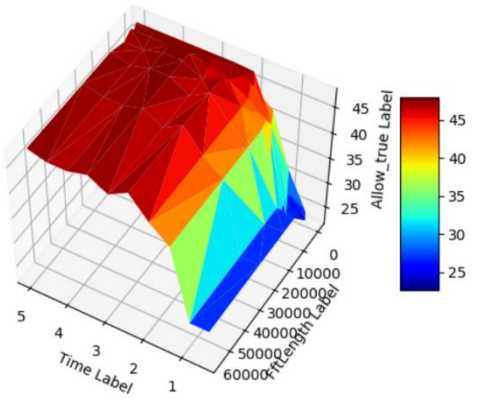


Рисунок 3.26 – Графік успішного проходження зі злодієм

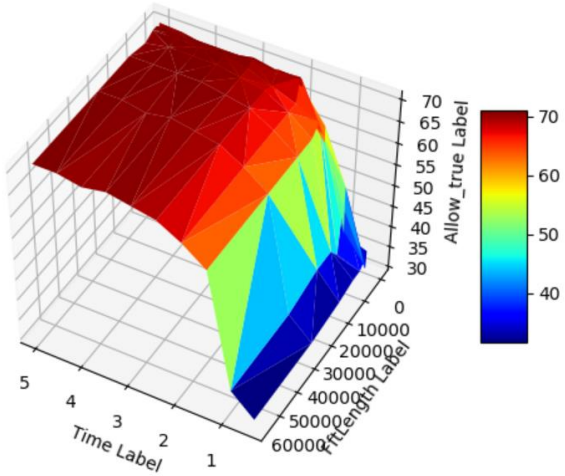


Рисунок 3.27 – Графік успішного проходження без злодія

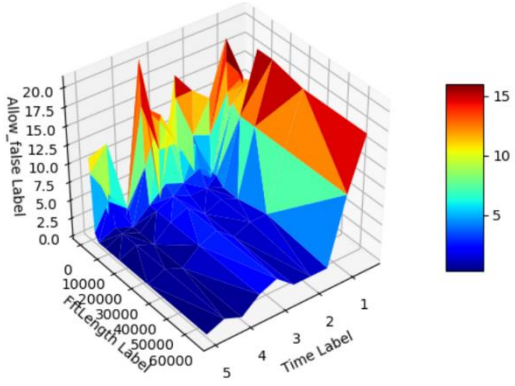


Рисунок 3.28 – Графік не успішного проходження зі злодієм

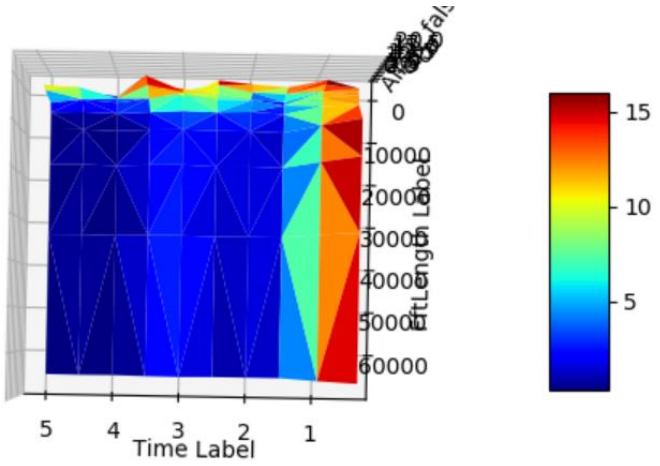


Рисунок 3.29 – Графік не успішного проходження без злодія

Знову ж таки, що чим більше FFT та тривалість, то більша ефективність. Але коли FFT ≥ 8192, то його ефективність не особливо впливає на результат. Тому краще брати FFT 8192.

По експерименту можна помітити, що тривалість від 4 до 5 є найефективнішою та є постійною. Тому краще всього взяти тривалість 4,5.

Тепер потрібно провести експеримент, де FFT=8192, тривалість 4,5, а нижній поріг – 2.

У таблиці 3.3 показано результат зі злодіями з цими даними, де timeToSplit – тривалість фраз, на які розбивається його говоріння.

Таблиця 3.5 – Результат з злодіями

timeToSplit	allowTrue	allowFalse	allCount
4.5	28	0	30

Можна зробити висновок, що результат хороший, тому що з 30 зразків голосів, воно визначило 28 голосів правильно.

Також треба переглянути систему на правильність аутентифікації без злодіїв. У таблиці 3.4 показано результат.

Таблиця 3.6 – Результат без злодіїв

timeToSplit	allowTrue	allowFalse	allCount
4.5	25	0	30

Також можемо побачити висновок, що результат хороший.

Як говорилось в минулому підрозділі, також є додаток на телефоні. Він виводить мел-кепстральні коефіцієнти.

Там усі стани моделі можуть бути досягнутими з іншого стану лиш за один крок. Вона має таку властивість, що кожен коефіцієнт a_{ij} є позитивний. Обрано тип HMM такий, тому що стани, які будуть розробляти мел-кепстральні коефіцієнти, не повинні бути упорядковані по часовій осі, тому що так є при розпізнаванні за фонемами і там найчастіше використовується модель «зліва-направо».

Параметри задаються конструктором HMM. Параметри A, B та π розраховуються в процесі навчання. Інші параметри, такі як розмір алфавіту та кількість станів є вільними. Мел-кепстральні коефіцієнти є цілочисельними і є в діапазоні [0, 15]. Через це алфавіт моделі не може бути меншим, ніж 16 символів. Тому що він буде робити обрахунки складнішими та повільними.

На рисунку 3.30 показано схему обирання параметрів моделі HMM.

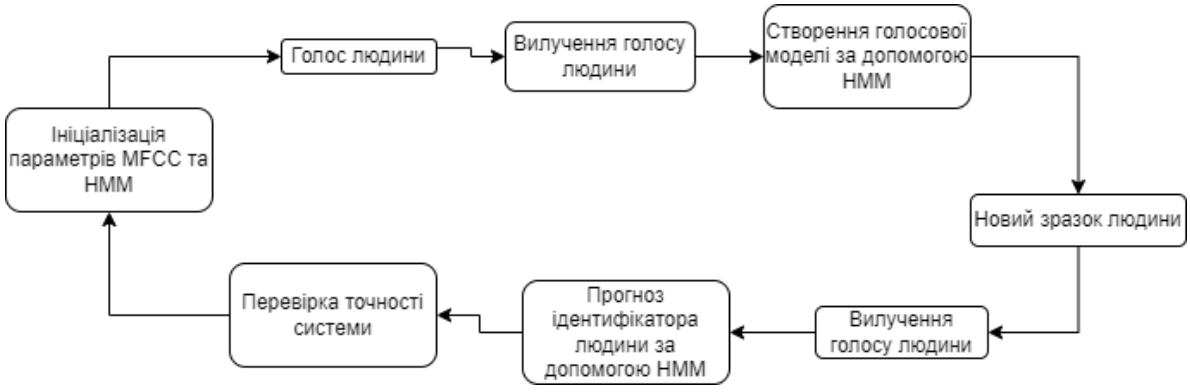


Рисунок 3.30 – Схема обирання параметрів моделі HMM

За допомогою цієї схеми можна обирати правильну кількість прихованих станів та мел-кепстральних коефіцієнтів. Перших 3 коефіцієнти є дуже різноманітними, що різниця аж може бути більше 2 для однієї і тієї самої людини, тому це дуже негативно впливає на ідентифікацію. А коефіцієнти 25 і більше є стабільними, що теж негативно впливає. На рисунку 3.31 показано скріншот програми, яка показує мел-кепстральні коефіцієнти голосу.

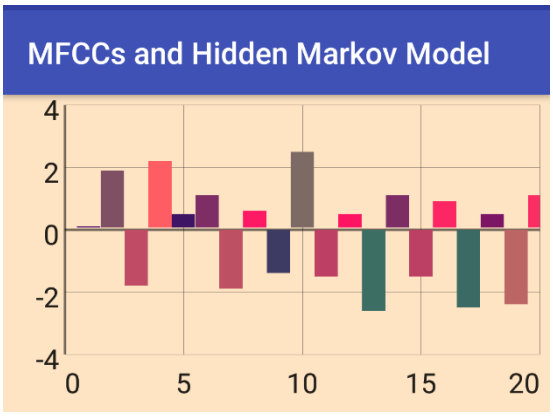


Рисунок 3.31 – Скріншот програми, яка показує мел-кепстральні коефіцієнти голосу

3.5 Висновки

У цьому розділі, було обговорено, як створювався проект. Було показано, як виглядає пристрій та його з'єднання. Також було показано за допомогою блок-схем, як працює додаток, як він виглядає та його тестування.

ВИСНОВКИ

Розумний дім – це дуже зручно. Тому що не потрібно багато про що піклуватись, наприклад, що ліньки піти увімкнути світло, чи відкрити штори. Але також зручно те, що можна зробити так, що до будинку можна потрапити тільки завдяки вашому голосу, що є дуже круто. Це гарантує безпеку та комфорт.

Метою цього проекту було проектування та створення приладу, який буде проводити аутентифікацію за голосом для того, щоб дозволити чи заборонити вхід до розумного будинку.

У першому розділі було обговорено про підгрупу для розпізнавання голосу – голосова біометрія, яка дуже важлива і корисна для розробки даного проекту для розумного будинку. Також про мінуси та плюси голосової аутентифікації та її актуальності. І було розібрано, як воно все працює та які методи можна застосувати для розробки.

У другому розділі був аналіз компонентів, які потрібні для апаратної частини приладу та що потрібно для програми для ідентифікації голосу для розумного будинку.

І в останньому було розглянуто, як виглядає схема і збірка приладу, який принцип роботи всієї аутентифікації правильної людини для входу у будинок. І також було показано, який був розроблений додаток та його тестування при тому, коли є злодії і без них.

Цінність цього проекту є в тому, що цей пристрій можна використовувати та поєднати з іншими функціями в розумному будинку, по типу виконання команд в будинку чи навіть в саду.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Система розпізнавання голосу за допомогою мікроконтролера. URL: https://microcontrollerslab.com/voice-recognition-system-using-microcontroller/ (дата звернення: 04.03.2022).
2. Bergeron B. Voice recognition for Robotic Control. URL: https://www.crustcrawler.com/articles/Voice-Recognition.pdf (дата звернення: 30.03.2022).
3. Voice Recognition. URL: https://www.piddlerintheroot.com/voice-recognition/ (дата звернення: 02.04.2022).
4. Muhammad N.K. Biometric Voice Recognition in Security System. Melaka. 2014. pp. 104-106.
5. Arduino Voice Recognition Module. URL: https://www.geeetech.com/wiki/index.php/Arduino_Voice_Recognition_Module (дата звернення: 03.04.2022).
6. Voice commands. URL: https://www.ardumotive.com/how-to-use-a-voice-recognition-module.html (дата звернення: 04.05.2022).
7. Бідюк П., Бондарчук В. Сучасні методи біометричнох ідентифікації. Київ: ІПСА, 2009. 142 с.
8. Дубчак О. В., Максімов Ю.О. Аналіз ефективності та надійності методів біометричної аутентифікації. URL: http://www.rusnauka.com/1_NIO_2011/Medecine/77655.doc.htm (дата звернення: 14.03.2022).
9. Засоби аутентифікації за голосом. URL: https://www.evkova.org/referaty/sredstva-autentifikatsii-po-golosu (дата звернення: 12.03.2022).
10. Кухарєв Г.А. Біометричні системи: методи та засоби ідентифікації особистості людини. Харків, 2001. 240 с.

11. Ian M., Hamid R.S. Speech Recognition for Smart Homes. Singapore. 2008. pp. 479-482.

12. Shrawankar U., Thakare V.M. Techniques for feature extraction in speech recognition nsystem: a comparative study. IJCAETS. 2013. pp. 412–418.

13. Stuttle M., Gales M.J. A mixture of gaussians front end for speech recognition. *Seventh European Conference on Speech Communication and Technology*. 2001. Sep 3–7. Aalborg, Denmark. 2001. pp. 675–678.

14. Голубєв Г. А., Габрієлян Б. А. Сучасний стан та перспективи розвитку біометричних технологій. 10 видання, Херсон, 2004. С. 39 – 46.

15. Рабінер Л.Р., Шафер Р.В. Цифрова обробка мовних сигналів. *М: Радіо і зв'язок*, 1981. 495 с.

16. Добрушкін Г. О., Данилов В. Я. Основні підходи до розпізнавання мовленнєвої інформації. Вінниця: ВПІ, 2010. С. 61-73.

17. Zhong Y., Raman T., Burkhardt C., Jeffrey P. JustSpeak: Enabling Universal Voice Control on Android. 2014. pp. 33-34.

18. Perera C. URL: https://www.mdpi.com/1424-8220/22/4/1325/htm (дата звернення: 14.04.2022).

19. Lama P., Namburu M. Speech recognition with dynami time warping using MATLAB. 2010. p. 21.

20. Jayalakshmamma R., Naganjaneyulu P. V., Babulu K. Implementation of Integrity of Voice and Face Recognition for Home Security By Using Gsm and Zigbee. n4, 2012. pp. 1043–1047.

21. Ishengoma F.R. Authentication System for Smart Homes Based on ARM7TDMI-S and IRIS-Fingerprint Recognition Technologies. *Int. J. Inf. Eng. Electron. Bus*. 2014. Vol. 6, No. 6. pp. 64–69.

22. Speech Recognition Kit Construction Manual & User Guide URL: https://www.imagesco.com/speech/SR-07.pdf (дата звернення: 11.05.2022).

23. Sudharsan B., Corcoran P., Muhammad A. I. Smart speaker design and implementation with biometric authentication and advanced voice interaction capability. Galway, 2019. pp. 3-4.

24. Biometrics A. Mobile biometric authentication framework - aware biometrics. URL: https://www.aware.com/knomi-mobile-biometric-authentication/ (дата звернення: 10.05.2022).

25. Biometrics A. Voice authentication technology - aware biometrics software. URL: https://www.aware.com/voice-authentication/ (дата звернення: 12.05.2022).

26. Add Voice Control to Your Smart Home Devices. URL: https://www.smarthome.com/blogs/tips-tricks/add-voice-control-to-your-smart-home-devices (дата звернення: 02.03.2022).

27. SmartHome system. URL: http://smarthome.geekster.com (дата звернення: 16.03.2022).

28. Rabiner L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 1998. pp. 257-286.

29. Mantoro T., Adnan M. A., Ayu M. A. Secured Communication between Mobile Devices and Smart Home Appliances. *Int. Conf. Adv. Comput. Sci. Appl. Technol*, 2013. pp. 429-434.

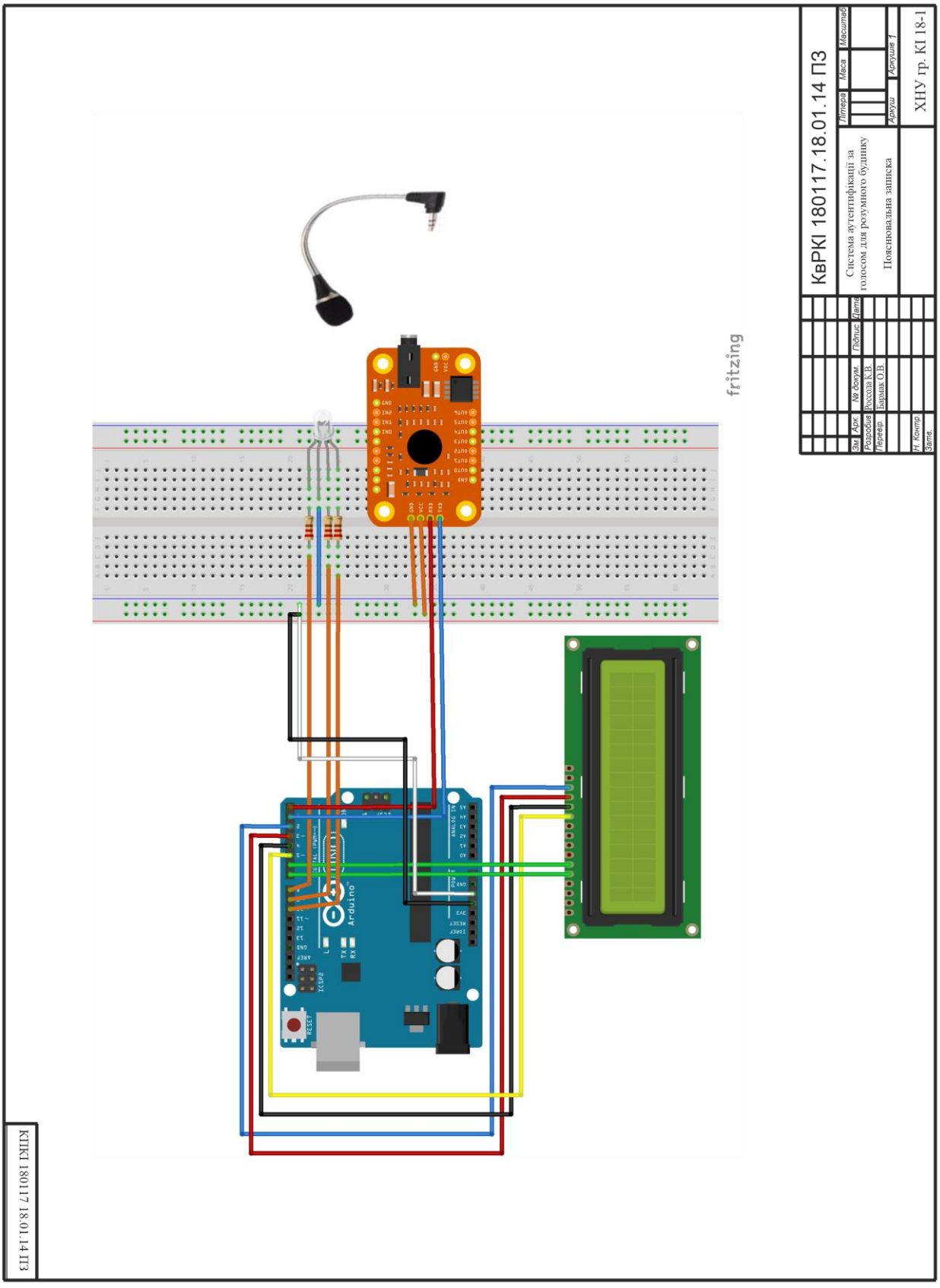
30. How Consumers Are Adapting To Voice Technology In The Smart Home. URL: https://www.kardome.com/blog-posts/consumers-adapting-voice-technology-smart-home#:~:text=Voice%20control%20is%20the%20primary,speaker%20use%20their%20device%20daily. (дата звернення: 24.03.2022).

31. Voice recognition (speaker recognition). URL: https://www.techtarget.com/searchcustomerexperience/definition/voice-recognition-speaker-recognition#:~:text=Voice%20or%20speaker%20recognition%20is,Apple's%20Siri%20and%20Microsoft's%20Cortana. (дата звернення: 08.03.2022).

Додаток А

(обов’язковий)

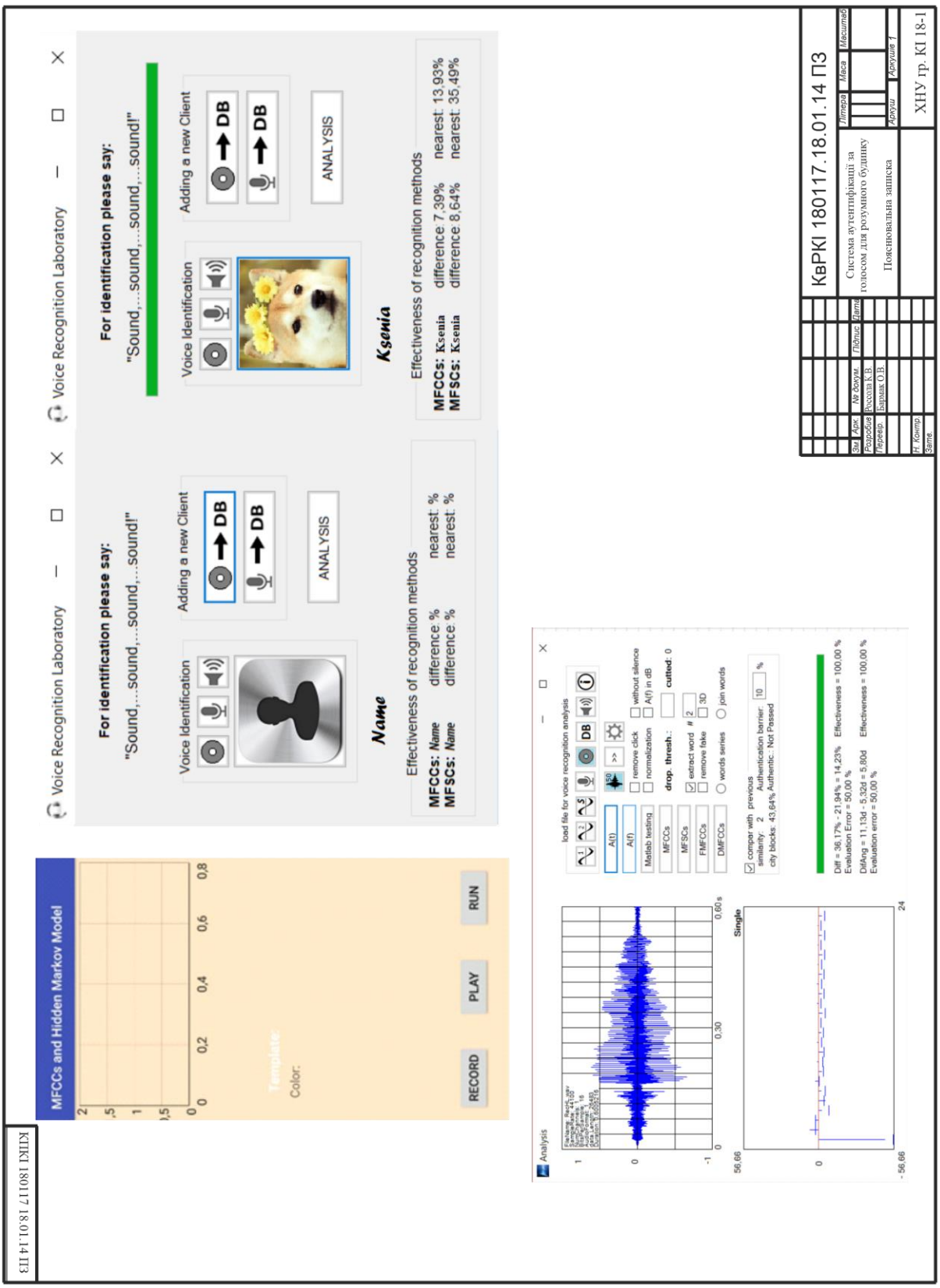
Копія креслення «Схема апаратних з’єднань»



Додаток Б

(обов’язковий)

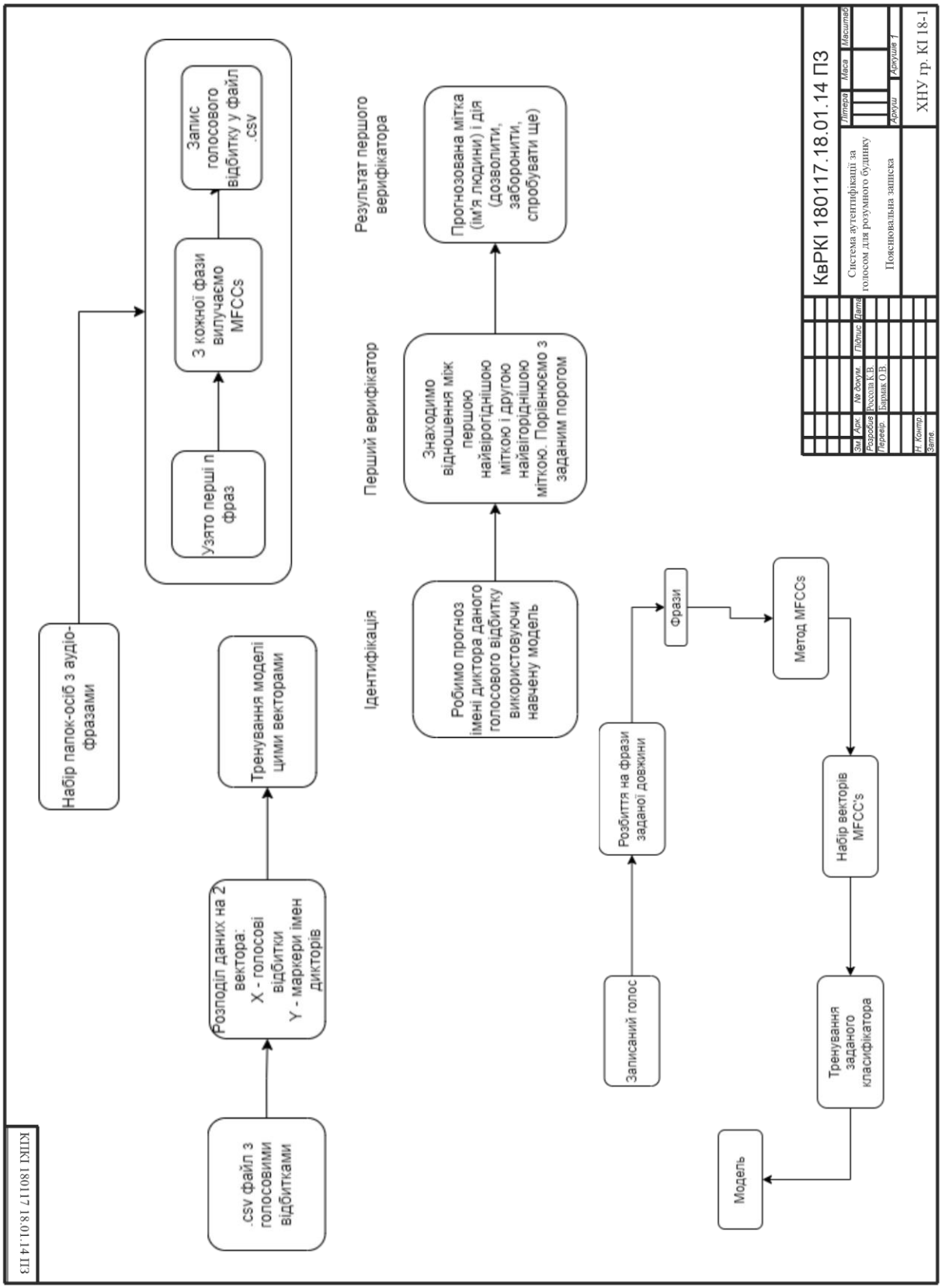
Копія креслення «Інтерфейс програми»



Додаток В

(обов’язковий)

Копія креслення «Блок-схеми програми»



Додаток Г

(обов’язковий)

Лістинг коду

read_text.py:

```
# for recording:
from threading import Timer
# for random phrases:
import random

text_file_name = 'texts/ukraine_history.txt'

def read_random_text(number_of_rows):
    f = open(text_file_name, "r", encoding="utf-8")

    # read all lines
    lines = f.readlines()

    # choise random line
    rand_line = random.randint(0,len(lines)-number_of_rows-1) # this should make
it work
    end_line = rand_line+number_of_rows

    text_to_output = ''
    # print lines
    while rand_line < end_line:
        text_to_output+=lines[rand_line]
        rand_line += 1

    return text_to_output

# print(read_random_text(6))
```

text_into_phrases.py:

```
# for reading .wav-file
import scipy.io.wavfile as wav
from pydub import AudioSegment
# for work with directories
import os
# for remove directory
import shutil

def delete_folder_content(folder):
    # folder = '/path/to/folder'
    try:
        for the_file in os.listdir(folder):
            file_path = os.path.join(folder, the_file)
            try:
                if os.path.isfile(file_path):
                    os.unlink(file_path)
                elif os.path.isdir(file_path): shutil.rmtree(file_path)
            except Exception as e:
```

```
            print(e)
    except Exception as e:
            print(e)


def split_file(directory_from, PERSONS_DIRECTORY_TO, filename, time):
    # full file path
    fullFilenamePath = ''

    if directory_from=='':
        fullFilenamePath = filename
    else:
        fullFilenamePath = directory_from+'/'+filename

    (rate,sig) = wav.read(fullFilenamePath)
    audio = AudioSegment.from_wav(fullFilenamePath)
    #splice frames to get a list strings each representing a 'time' length
    #wav file

    # duration of whole .wav-file
    duration = len(sig)/rate
    durationMilisec = duration*1000
    # time in miliseconds
    timeMilisec = time*1000
    x=0

    # let's find index of char that is near .wav
    index = filename.find('.wav')
    directory = filename.replace('.wav','')
    directory = PERSONS_DIRECTORY_TO+'/'+directory

    if not os.path.exists(directory):
        os.makedirs(directory)

    while x+timeMilisec<=durationMilisec:
        # nea audio frame
        newAudio= audio[x:x+timeMilisec]
        # create newAudio filename
        newAudio_filename = directory+'/'+filename[:index] + str(int(x)) +
filename[index:]
        #Exports to a wav file in the current path.
        newAudio.export(newAudio_filename, format="wav")
        # iterate x
        x=x+timeMilisec


# all persons in main dirname
def split_all_files(PERSONS_DIRECTORY_FROM, PERSONS_DIRECTORY_TO, TIME_TO_SPLIT):
    # # PERSONS_DIRECTORY_TO = 'real_voices'
    # try:
    #     # delete old directory
    #     shutil.rmtree(PERSONS_DIRECTORY_TO, ignore_errors=True)
    # except:
    #     print("An exception occurred")

    delete_folder_content(PERSONS_DIRECTORY_TO)
    # create new directory
    if not os.path.exists(PERSONS_DIRECTORY_TO):
        os.makedirs(PERSONS_DIRECTORY_TO)

    # PERSONS_DIRECTORY_FROM = 'real_voices_texts'
```

```
    # TIME_TO_SPLIT = 1.0

    all_persons = os.listdir(PERSONS_DIRECTORY_FROM)
    for filename in  all_persons:
        split_file(PERSONS_DIRECTORY_FROM, PERSONS_DIRECTORY_TO, filename,
TIME_TO_SPLIT)
```

header.py:

```
# for voice recording: >>>>>>>>>>>>>>>>>>>>>>>>
import sys
from sys import byteorder
from array import array
from struct import pack

import pyaudio
import wave
# for voice recording <<<<<<<<<<<<<<<<<<<<<<<<
# ABOUT AUDIO FILE >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
THRESHOLD = 500
CHUNK_SIZE = 1024
# mfccs parameters:
# by default
FFT_LENGTH = 8192
# FFT_LENGTH = 65536  #IN BACHLORE WORK
FORMAT = pyaudio.paInt16
RATE = 44100
# ABOUT AUDIO FILE <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
from python_speech_features import mfcc
import scipy.io.wavfile as wav


# for testing KNeighbours
# for array processing
import numpy as np
# # for graphs
# import matplotlib.pyplot as plt
# to work with .csv/txt/etc.
import pandas as pd
# Import LabelEncoder
from sklearn import preprocessing

#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# from sklearn.metrics.pairwise import manhattan_distances

# to save model on disk
from joblib import dump, load
# # for clustering

# Local Outlier Factor (LOF) - Neighbours:
from sklearn.neighbors import LocalOutlierFactor

# for random text:
from read_text import *

# import text_into_phrases
from text_into_phrases import *
```

```
# additional parameters from bachloar work >>>>>>
numcep = 24
lowfreq = 20
highfreq = 8000
# from bachlor work <<<<<<<<<<<<<<<<<<<<<<<<<<<<<

# temporary directory for persons
temp_directory_for_persons = 'temp_directory_for_phrases'
# directory with big textes
PERSONS_DIRECTORY_FROM = 'real_voices_texts'
```

writeToDb.py:

```
# IMPORT HEADER FROM HEADER.PY
from header import *

# for work with directories
import os
# for .flac to .wav converting
from pydub import AudioSegment

# import text_into_phrases
from text_into_phrases import *

# for random text:
from read_text import *

# about audio file >>>>>>>>:
DURATION_of_all_phrase = 20
TIME_TO_SPLIT = 4.5
# about audio file <<<<<<<<


# ABOUT DATASET>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
# dataset directory
dataset_directory = 'real_voices'

# # number of persons to learn (at all = 40):
# number_of_persons_to_learn = 5

# minimal number of records per each person = 55

# sequence number of record from start FROM
# ///////////////////////
number_to_learn_FROM = 0
# ///////////////////////

# number of records to learn:
number_to_learn = 3
# sequence number of record to learn TO
number_to_learn_TO = number_to_learn_FROM+number_to_learn

# avout dataset <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

# temporary directory for phrases
temp_directory_for_phrases = 'temp_directory_for_phrases/demo/'

# VOICE-FILE RECORDING:>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
# timer>>>>>:
# isSilent flag. whn thiflag is silent, than recording is stop
letsStop = False

def timeout():
    global letsStop
    letsStop=True

    # duration is in seconds
t = Timer(DURATION_of_all_phrase, timeout)
# timer<<<<<<
def is_silent(snd_data):
    "Returns 'True' if below the 'silent' threshold"
    # let's make it not truncate when it is silent
    return max(snd_data) < THRESHOLD
    # return isSilent

def normalize(snd_data):
    "Average the volume out"
    MAXIMUM = 16384
    times = float(MAXIMUM)/max(abs(i) for i in snd_data)

    r = array('h')
    for i in snd_data:
        r.append(int(i*times))
    return r

def trim(snd_data):
    "Trim the blank spots at the start and end"
    def _trim(snd_data):
        snd_started = False
        r = array('h')

        for i in snd_data:
            if not snd_started and abs(i)>THRESHOLD:
                snd_started = True
                r.append(i)

            elif snd_started:
                r.append(i)
        return r

    # Trim to the left
    snd_data = _trim(snd_data)

    # Trim to the right
    snd_data.reverse()
    snd_data = _trim(snd_data)
    snd_data.reverse()
    return snd_data

def add_silence(snd_data, seconds):
    "Add silence to the start and end of 'snd_data' of length 'seconds' (float)"
    r = array('h', [0 for i in range(int(seconds*RATE))])
    r.extend(snd_data)
    r.extend([0 for i in range(int(seconds*RATE))])
    return r

def record():
    """
    Record a word or words from the microphone and
    return the data as an array of signed shorts.
```

```
    Normalizes the audio, trims silence from the
    start and end, and pads with 0.5 seconds of
    blank sound to make sure VLC et al can play
    it without getting chopped off.
    """
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT, channels=1, rate=RATE,
        input=True, output=True,
        frames_per_buffer=CHUNK_SIZE)

    num_silent = 0
    snd_started = False

    r = array('h')

    while 1:
        # little endian, signed short
        snd_data = array('h', stream.read(CHUNK_SIZE))
        if byteorder == 'big':
            snd_data.byteswap()
        r.extend(snd_data)

        silent = is_silent(snd_data)

        if silent and snd_started:
            num_silent += 1
        elif not silent and not snd_started:
            snd_started = True
            # start timer: >>>
            t.start()
            # start timer <<<<

        # if snd_started and num_silent > 30:
        #     break
        if letsStop:
            break

    sample_width = p.get_sample_size(FORMAT)
    stream.stop_stream()
    stream.close()
    p.terminate()

    r = normalize(r)
    r = trim(r)
    # r = add_silence(r, 0.5)
    return sample_width, r

def record_to_file(path):
    "Records from the microphone and outputs the resulting data to 'path'"
    sample_width, data = record()
    data = pack('<' + ('h'*len(data)), *data)

    wf = wave.open(path, 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(sample_width)
    wf.setframerate(RATE)
    wf.writeframes(data)
    wf.close()
# VOICE-FILE RECORDING:<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


if __name__ == '__main__':
```

```
# 3 REAL PERSONS VOICE RECORDING PART :
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
# first sample in Db:
   # create dataset.csv with mfccs header
    with open('demo_dataset.csv','w') as file:
        # strLine = 'person, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13'
        strLine = 'person, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 
17, 18, 19, 20, 21, 22, 23, 24'
        file.write(strLine)
        file.write('\n')
    # let's split voices:
    split_all_files(PERSONS_DIRECTORY_FROM, dataset_directory, TIME_TO_SPLIT)
    # extract all person's (folders)
    persons_to_learn = os.listdir(dataset_directory)
    # let's write each person to Db
    for person_dir in persons_to_learn:
        # extract full person directory path
        full_person_dir = os.path.join(dataset_directory, person_dir)
        # all audio-files in current person's folder
        files = [os.path.join(full_person_dir,f) for f in 
os.listdir(full_person_dir) if os.path.isfile(os.path.join(full_person_dir, f))]
        # selected audio files from person's folder
        files_new = files[:number_to_learn]

        # let's foreach every file from person's folder
        for demo in files_new:
            # PUT THEIR NAME OF CURRENT INDIVIDUAL SAMPLE
            LABEL = person_dir
            # # for DICTORS: >>>>>>>>>>>>>>>>>>>>>>>>>
            #     # let's convert .flac to .wav
            # demo = AudioSegment.from_file(demo, "flac")

            # # convert to .wav
            # demo.export("demo.wav", format="wav")
            # (rate,sig) = wav.read("demo.wav")
            # # for DICTORS <<<<<<<<<<<<<<<<<<<<<<<<<

            # for REAL VOICES: >>>>>>>>>>>>>>>>>>>>
            # read .wav file
            (rate,sig) = wav.read(demo)
            # extract mfccs from demo.wav
            # for REAL VOICES <<<<<<<<<<<<<<<<<<<<

            # EXTRACT MFCCs FROM SIG, RATE:
            mfcc_feat = mfcc(sig,rate,winlen=0.094,nfft=FFT_LENGTH, numcep=numcep, 
lowfreq=lowfreq, highfreq=highfreq) #bachlor parameter

            #let's dump mfccs string to .csv:
            with open('demo_dataset.csv','a') as file:
                for line in mfcc_feat:
                    strLine = str(LABEL) + ',' + ','.join(map(str, line))
                    file.write(strLine)
                    file.write('\n')
        # message to terminal after every person is recorded to DB-file
        print("mfccs of "+str(person_dir)+" was recorded to .csv")

# 3 REAL PERSONS VOICE RECORDING PART 
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

# 
==============================================================================
=======
```

```
# # REAL PERSON VOICE RECORDING PART:
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
# #  recording part
#     # print("please, enter your name:")
#     # label = input("")
#     # PUT THEIR NAME OF CURRENT INDIVIDUAL SAMPLE
#     # LABEL = "y"
#     # print("ваше ім'я:" + str(LABEL))
#     LABEL = input("введіть першу букву свого імені (на англійській): ")

#
print('================================================================================
=======')
#     print("прочитайте фразу в мікрофон поки не вісвітиться повідомлення (20
сек):")
#
print('================================================================================
=======')

#     print(read_random_text(10))
#     # print(str(random_sentence))
#     record_to_file('demo.wav')

#
print('================================================================================
=======')
#
print('================================================================================
=======')
#
print('================================================================================
=======')
#
print('================================================================================
=======')
#     print('==================================ФРАЗА
ЗАПИСАНА!==================================')
#
print('================================================================================
=======')
#
print('================================================================================
=======')
#
print('================================================================================
=======')
#
print('================================================================================
=======')

#     # # COMMENT CODE BELOW IF THIS IS NOT FIRST RECORD FOR THIS DB: >>>>>>>>>
#     # # create dataset.csv with mfccs header
#     # with open('demo_dataset.csv','w') as file:
#     #     # strLine = 'person, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13'
#     #     strLine = 'person, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24'
#     #     file.write(strLine)
#     #     file.write('\n')
#     # # COMMENT CODE ABOVE IF THIS IS NOT FIRST RECORD FOR THIS DB: <<<<<<<<<


#     # let's delete directory:
#     delete_folder_content(temp_directory_for_persons)
```

```
#     # let's split voice to smaller audio files:
#     split_file('',temp_directory_for_persons, 'demo.wav', TIME_TO_SPLIT)

#      # all audio-files in current person's folder
#     files = [os.path.join(temp_directory_for_phrases,f) for f in
os.listdir(temp_directory_for_phrases) if
os.path.isfile(os.path.join(temp_directory_for_phrases, f))]

#     files_to_learn = files[:number_to_learn]

#     for file_phrase in files_to_learn:
#         (rate,sig) = wav.read(file_phrase)
#         mfcc_feat = mfcc(sig,rate,winlen=0.094,nfft=FFT_LENGTH, numcep=numcep,
lowfreq=lowfreq, highfreq=highfreq) #bachlor parameter

#         # st sample in Db:
#         #text=List of strings to be written to file
#         with open('demo_dataset.csv','a') as file:
#             for line in mfcc_feat:
#                 strLine = str(LABEL) + ',' + ','.join(map(str, line))
#                 file.write(strLine)
#                 file.write('\n')

#     print("mfccs було записано в .csv")
# # REAL PERSON VOICE RECORDING PART
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

learning.py:

```
# for array processing
import numpy as np
# to work with .csv/txt/etc.
import pandas as pd
# Import LabelEncoder
from sklearn import preprocessing
# Split the data between the Training Data and Test Data
from sklearn.model_selection import train_test_split
# Classifier:
# kneigbours:
from sklearn.neighbors import KNeighborsClassifier
# Gradient Boosting:
# from sklearn.ensemble import GradientBoostingClassifier
# forest:
# from sklearn.ensemble import RandomForestClassifier
# # ada boost:
# from sklearn.ensemble import AdaBoostClassifier

# to save model on disk
from joblib import dump, load

# read DB-file with mfccs:
# header = None if dataset has no header
dataset =  pd.read_csv('demo_dataset.csv',sep=',')

# let's encode catigorial data to numbers
dataset_encoded = dataset.copy()
# creating labelEncoder
le = preprocessing.LabelEncoder()
```

```
# Converting string labels into numbers.
dataset_encoded['person']=le.fit_transform(dataset['person'].astype(str))
# let's save LabelEncoder on disk
dump(le, 'LabelEncoder.joblib')

# make 2 datasets X - for input data and Y- for output
#  input data
# X = dataset_encoded.iloc[:, 1:14].values  #for default mfccs parameters
X = dataset_encoded.iloc[:, 1:25].values # for bachloar mfccs parametrs

#  output data
Y = dataset_encoded.iloc[:, 0].values

# CLASSIFIER: >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

# KNeighbours >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
clf = KNeighborsClassifier(n_neighbors=3)

# << KNeighbours <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
# clf = GradientBoostingClassifier(loss='deviance', learning_rate=0.1,
n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3,
min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None,
max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False,
presort='auto', validation_fraction=0.1, n_iter_no_change=None, tol=0.0001)
# let's train:
clf.fit(X, Y)
# xgboost:<<<<<<<<<<<<<<<


# let's save model on disk
dump(clf, 'Classifier_model.joblib')
# CLISSIFIER <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

authentication.py:

```
# let's import all other imports
from header import *

# about audio file: >>>>>>>>
DURATION_of_all_phrase = 6
TIME_TO_SPLIT = 4.5
# about audio file: <<<<<<<<

# temporary directory for phrases
temp_directory_for_phrases = 'temp_directory_for_phrases/on_inspection/'

# VOICE-FILE RECORDING:>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
# timer>>>>>:
# isSilent flag. whn thiflag is silent, than recording is stop
letsStop = False

def timeout():
    global letsStop
    letsStop=True

    # duration is in seconds
t = Timer(DURATION_of_all_phrase, timeout)
```

```
# timer<<<<<<
def is_silent(snd_data):
    "Returns 'True' if below the 'silent' threshold"
    # let's make it not truncate when it is silent
    return max(snd_data) < THRESHOLD
    # return isSilent

def normalize(snd_data):
    "Average the volume out"
    MAXIMUM = 16384
    times = float(MAXIMUM)/max(abs(i) for i in snd_data)

    r = array('h')
    for i in snd_data:
        r.append(int(i*times))
    return r

def trim(snd_data):
    "Trim the blank spots at the start and end"
    def _trim(snd_data):
        snd_started = False
        r = array('h')

        for i in snd_data:
            if not snd_started and abs(i)>THRESHOLD:
                snd_started = True
                r.append(i)

            elif snd_started:
                r.append(i)
        return r

    # Trim to the left
    snd_data = _trim(snd_data)

    # Trim to the right
    snd_data.reverse()
    snd_data = _trim(snd_data)
    snd_data.reverse()
    return snd_data

def add_silence(snd_data, seconds):
    "Add silence to the start and end of 'snd_data' of length 'seconds' (float)"
    r = array('h', [0 for i in range(int(seconds*RATE))])
    r.extend(snd_data)
    r.extend([0 for i in range(int(seconds*RATE))])
    return r

def record():
    """
    Record a word or words from the microphone and
    return the data as an array of signed shorts.

    Normalizes the audio, trims silence from the
    start and end, and pads with 0.5 seconds of
    blank sound to make sure VLC et al can play
    it without getting chopped off.
    """
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT, channels=1, rate=RATE,
        input=True, output=True,
        frames_per_buffer=CHUNK_SIZE)
```

```
    num_silent = 0
    snd_started = False

    r = array('h')

    while 1:
        # little endian, signed short
        snd_data = array('h', stream.read(CHUNK_SIZE))
        if byteorder == 'big':
            snd_data.byteswap()
        r.extend(snd_data)

        silent = is_silent(snd_data)

        if silent and snd_started:
            num_silent += 1
        elif not silent and not snd_started:
            snd_started = True
            # start timer: >>>
            t.start()
            # start timer <<<<

        # if snd_started and num_silent > 30:
        #     break
        if letsStop:
            break

    sample_width = p.get_sample_size(FORMAT)
    stream.stop_stream()
    stream.close()
    p.terminate()

    r = normalize(r)
    r = trim(r)
    # r = add_silence(r, 0.5)
    return sample_width, r

def record_to_file(path):
    "Records from the microphone and outputs the resulting data to 'path'"
    sample_width, data = record()
    data = pack('<' + ('h'*len(data)), *data)

    wf = wave.open(path, 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(sample_width)
    wf.setframerate(RATE)
    wf.writeframes(data)
    wf.close()
# VOICE-FILE RECORDING:<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<



#
========================+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+
# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>Varification
    # check if X_inspect and  X_pred belongs to one cluster
def check_by_Verificator(X_inspect, Y_inspect, X_pred, Y_pred, result_individual,
IDENT_flag):

    novelty_relation = 0
    #  Local Outlier Factor (LOF)
    # novelty = True / it is to make novelty prediction.
```

```
    # when you train clean data and check it with anomaly on new data
    clf = LocalOutlierFactor(n_neighbors=20, algorithm='auto', leaf_size=30,
metric='minkowski', p=2, metric_params=None,contamination='legacy', novelty=True,
n_jobs=None)

    # let's fit with predict X-value
    clf.fit(X_pred)
    # let's predict with inspect X-value
    Y_clf_pred = clf.predict(X_inspect)
    unique, counts = np.unique(Y_clf_pred, return_counts=True)
    print(dict(zip(unique, counts)))
    # (-1)/1   novelty_relation = not_equels_count/equels_count
    if(len(counts)>1):
        novelty_relation = counts[1]/counts[0]
    else:
        novelty_relation = 9999

    print('NOVELTY_RELATION: '+str('%.3f'%(novelty_relation)))
    if novelty_relation > 0.68:
        VARIFY_flag = 2
    else:
        if novelty_relation < 0.4:
            VARIFY_flag = 0
        else:
            VARIFY_flag = 1


# when main method is one of both:
    if(IDENT_flag==2):
        action = "ALLOW"
    else:
        if(IDENT_flag==0):
            action = "DENY"
        # if IDENT_flag == TRY MORE
        else:
            if VARIFY_flag==2:
                action = "ALLOW"
            else:
                if VARIFY_flag==0:
                    action = "DENY"
                # if IDENT_flag == TRY MORE and VARIFY_flag==TRY MORE
                else:
                    action = "TRY MORE"

    # print("action: "+str(action))

#
////////////////////////////////////////////////////////////////////////////////
/////////////////
    return result_individual, action
#
==============================++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
# <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Varification
if __name__ == '__main__':
    # recording part
    label = "on_inspection"


print('==========================================================================
=======')
    print("прочитайте фразу в мікрофон поки не вісвітиться повідомлення (6 сек):")
```

```
print('================================================================================
=======')

    print(read_random_text(10))

    record_to_file('on_inspection.wav')


print('================================================================================
=======')

print('================================================================================
=======')

print('================================================================================
=======')

print('================================================================================
=======')
    print('==================================ФРАЗА
ЗАПИСАНА!==================================')

print('================================================================================
=======')

print('================================================================================
=======')

print('================================================================================
=======')

print('================================================================================
=======')

    # let's delete directory:
    delete_folder_content(temp_directory_for_persons)
    # let's split voice to smaller audio files:
    split_file('',temp_directory_for_persons, 'on_inspection.wav', TIME_TO_SPLIT)

    # all audio-files in current person's folder
    files = [os.path.join(temp_directory_for_phrases,f) for f in
os.listdir(temp_directory_for_phrases) if
os.path.isfile(os.path.join(temp_directory_for_phrases, f))]

    on_inspection = files[0]
    (rate,sig) = wav.read(on_inspection)
    mfcc_feat = mfcc(sig,rate,winlen=0.094,nfft=FFT_LENGTH, numcep=numcep,
lowfreq=lowfreq, highfreq=highfreq)
    # le's print results
    print('\n\n')

print('================================================================================
=')

print('==================================results:=======================================
=')

print('================================================================================
=')
    print('\n\n')

    print('DURATION: '+str(len(sig)/rate))
```

```
    ##text=List of strings to be written to file
    with open('on_inspection_dataset.csv','w+') as file:
        file.write('person, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24 \n')
        for line in mfcc_feat:
            strLine = str(label) + ',' + ','.join(map(str, line))
            file.write(strLine)
            file.write('\n')

    # LET'S TEST ON_INSPECTION SAMPLE
    # header = None if dataset has no header
    dataset =  pd.read_csv('on_inspection_dataset.csv',sep=',')

    # let's encode catigorial data to numbers
    dataset_encoded = dataset.copy()
    # creating labelEncoder
    le = preprocessing.LabelEncoder()
    # Converting string labels into numbers.
    dataset_encoded['person']=le.fit_transform(dataset['person'])

    # make 2 datasets X - for input data and Y- for output
    #  input data
   # X_inspect = dataset_encoded.iloc[:, 1:14].values # for default mfccs
parameters
    X_inspect = dataset_encoded.iloc[:, 1:25].values #for bachlors work mfccs
parametrs

    #  output data
    Y_inspect = dataset_encoded.iloc[:, 0].values

#
==============================++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
# Identification>>>>>>>>>>>>>>>>
    # kneigbours:>>>
    # load the learned model
    clf = load('Classifier_model.joblib')
    # let's predict
    # simple sci-learn clissifier
    y_pred = clf.predict(X_inspect)


    counts = np.bincount(y_pred)
    counts_sorted = np.argsort(counts, axis = 0)
    index_of_max_elem = counts_sorted[len(counts) - 1]
    index_of_second_max_elem = counts_sorted[len(counts) - 2]


    max_elem = counts[index_of_max_elem]
    second_max_elem = counts[index_of_second_max_elem]


    # let's count percentege between 1st max and 2nd max
    # must be from [1,9999]
    first_max_second_max_relation = max_elem/second_max_elem
    # ->>>the more so the better!

    # let's reverse from int to marks

    # load labelEncoder
    le = load('LabelEncoder.joblib')
    # result_mark = le.inverse_transform([max_freq_item])
    result_mark = le.classes_[index_of_max_elem]
```

```
    # global current_count
    IDENT_flag =0

    # three state: ALLOW(2)/TRY MORE(1)/DENY(0):
    if(first_max_second_max_relation < 2):
        IDENT_flag=0 #DENY
    if(2 <= first_max_second_max_relation and first_max_second_max_relation < 5):
        IDENT_flag=1 #TRY MORE
    if(first_max_second_max_relation>=5):
        IDENT_flag=2 #ALLOW

    # print('RELATION:'+ str(first_max_second_max_relation))
    #
==============================++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
    # Identification<<<<<<<<<<<<<<<

    # let's find all X_pred according to result_mark[0]
    demo_dataset = pd.read_csv('demo_dataset.csv')
    result_individual = result_mark
    pred = demo_dataset.loc[demo_dataset['person'].astype(str)==result_individual]
    # input data of predicted  individual
    # X_pred = pred.iloc[:, 1:14].values #for default mfccs parametrs
    X_pred = pred.iloc[:, 1:25].values #for mfccs parametrs from bachlor's work
    # output data of predicted  individual
    Y_pred = pred.iloc[:, 0].values
    #
==============================++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
    # Varification
    individual, action = check_by_Verificator(X_inspect, Y_inspect, X_pred,
Y_pred, result_individual, IDENT_flag)
    print('RELATION: '+ str('%.3f'%(first_max_second_max_relation)))
```

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1011328931

Дата перевірки:
25.05.2022 08:19:45 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
25.05.2022 08:23:40 EEST

ID користувача:
100005591

Назва документа: **Дипломна робота, Россола Система аутентифікації за голосом для розумного будинку**

Кількість сторінок: **56** Кількість слів: **7459** Кількість символів: **51918** Розмір файлу: **5.18 MB** ID файлу: **1011215073**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

2.17%
Схожість

Найбільша схожість: **0.66%** з джерелом з Бібліотеки (ID файлу: **1011215076**)

1.38% Джерела з Інтернету	20	Сторінка 58
1.06% Джерела з Бібліотеки	65	Сторінка 58

1.01% Цитат

Цитати	4	Сторінка 59

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	31
Підозріле форматування	12 сторінок

Wed May 25 07:30:19 EEST 2022, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверки: en_US, ru_RU, ua_UA.

Ошибок в документах: 7%

ID: 103922 Название: Система аутентифікації за голосом для розумного будинку Добавлено в БД: 2022-05-25 Авторы: К.В. Россола Руководители: О.В. Бармак Консультанты: Оппоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	43712	450	236 (1%)	4 (1%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Россола Ксенія Вікторівна

Тема: Система аутентифікації за голосом в розумному будинку

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 56

1. Короткий зміст роботи та прийнятих рішень: в рамках дипломного проекту було обговорено методи для голосової аутентифікації та створено прилад для цього в розумному будинку

2. Висновок про відповідність роботи дипломному завданню: дипломний проект у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині даного проекту

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: у першому, теоретичному, розділі дипломного проекту якісно та в повній мірі розглянуті методи вирішення поставленої задачі, був проаналізований кожен аспект, який стосується теми дипломного проекту. У другому розділі було розглянуто компоненти для створення приладу та створення програмної частини. У останньому, третьому розділі було розглянуто створення приладу, програми та тестування її. В загальному усі розділи відповідають завданню та містять сучасні методи вирішення поставлених завдань.

4. Позитивні сторони роботи: дипломний проект відповідає сучасним вимогам до проектування локальних комп'ютерних мереж та містить ряд інноваційних рішень, зокрема. Окремо можна виділити розглянуті методи, які використовуються для голосової аутентифікації, оскільки на сьогоднішній день таке рішення є досить затребуваним і має значні перспективи. Також важливим позитивним аспектом дипломного проекту є те, що прилад чудово може застосовуватись не тільки в розумному будинку.

5. Негативні сторони роботи: надмірна кількість теоретичного матеріалу, відсутність початкових налаштувань.

6. Оцінка графічного оформлення та пояснювальної записки роботи: пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному інженерно-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Манзюк Едуард Андрійович, к.т.н., доцент кафедри комп'ютерних наук _____

_____

" 31 " травня 2022 р. _____(підпис)

Завідувачу кафедри КІІС
д-ру техн.наук, проф. Говорущенко Т. О.

Россоли К.В.
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ-18-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

23.05.22
дата

підпис

87

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система голосової аутентифікації за голосом для розумного будинку

Автор: Россола Ксенія Вікторівна

Спеціальність: 123-Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Бармак Олександр Володимирович, д.т.н., професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, не є плагіатом, вони законні, тому що:

1) усі запозичені фрагментальні, або мають оформленні посилання;
2) всі зафіксовані зміни тексту відносяться до комбінування латинських символів з україномовними скороченнями і не є модифікацією тексту;
3) запозичення, які мають місце в розділах аналізу існуючих аналогів та прототипів, не описують безпосередньо авторське дослідження і не стосуються результатів роботи.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 2.17% і адресується до 85 першоджерел, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О. В. Бармак

Гарант ОП _____ С.М. Лисенко

Завідувач кафедри КІІС _____ Т.О. Говорущенко

88